R Graphics Cookbook

R数据可视化手册

[美] *Winston Chang* 著
肖楠 邓一硕 魏太云 译
邱怡轩 审校

O'REILLY®



R数据可视化手册

本书提供了快速绘制高质量图形的150多个精选的技巧，读者不需要了解R绘图系统的全部细节便可以掌握这些技巧。这里的每个技巧都针对一个具体问题。其中，讨论环节还解释了“如何”以及“为什么”如此绘图，读者可酌情将其套用到自己的项目中。

本书中的大多数方法使用的是以强大、灵活制图而著称的ggplot2包。如果读者对R语言有基本的了解，便可以开始学习R中丰富的数据可视化方法了。

- 使用R中的基础图形来快速探索数据
- 绘制各种柱状图、线图、散点图
- 用直方图、密度曲线、箱线图和其他图形来描述数据的分布
- 为图形添加注解，帮助读图者更好地理解数据
- 控制图形的整体外观
- 分组绘制图形以便于对比数据
- 绘图色彩的选择
- 绘制网络图、热图、3D散点图
- 把数据整理成绘图所需的格式

“这本书首先介绍了绘制常规图形的方法，然后展示了如何将它们调整为符合读者需要的图形。它不仅是R可视化方面的优秀读物，更以丰富的案例为我们提供了绘制图形的灵感源泉。”

——Hadley Wickham
莱斯大学助理教授

Winston Chang

RStudio的软件工程师，致力于R中的数据可视化和软件开发工具的研发。他创立的网站“Cookbook for R”提供了R中常见问题的解决技巧。

封面设计：Karen Montgomery，张健

O'Reilly Media, Inc.授权人民邮电出版社出版

此简体中文版仅限于中国大陆（不包含中国香港、澳门特别行政区和中国台湾地区）销售发行
This Authorized Edition for sale only in the territory of People's Republic of China (excluding Hong Kong, Macao and Taiwan)

分类建议：计算机/程序设计

人民邮电出版社网址：www.ptpress.com.cn



ISBN 978-7-115-34227-0

定价：89.00 元

R数据可视化手册

[美] *Winston Chang* 著

肖楠　邓一硕　魏太云　译

邱怡轩　审校

O'REILLY®

Beijing · Cambridge · Farnham · Köln · Sebastopol · Tokyo

O'Reilly Media, Inc.授权人民邮电出版社出版

人 民 邮 电 出 版 社

北　京

图书在版编目（CIP）数据

R数据可视化手册 /（美）常（Chang,W.）著；肖楠，邓一硕，魏太云译. -- 北京：人民邮电出版社，2014.5
ISBN 978-7-115-34227-0

Ⅰ. ①R… Ⅱ. ①常… ②肖… ③邓… ④魏… Ⅲ. ①可视化软件—手册 Ⅳ. ①TP31-62

中国版本图书馆CIP数据核字(2013)第307384号

版权声明

◆ 著　　　　[美] Winston Chang
译　　　　肖　楠　邓一硕　魏太云
审　校　　邱怡轩
责任编辑　杨海玲
责任印制　杨林杰
◆ 人民邮电出版社出版发行　北京市丰台区成寿寺路 11 号
邮编　100164　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京精彩雅恒印刷有限公司印刷
◆ 开本：800×1000　1/16
印张：21
字数：428 千字　　2014 年 5 月第 1 版
印数：1 – 4 000 册　　2014 年 5 月北京第 1 次印刷
著作权合同登记号　图字：01-2013-3677 号

定价：89.00 元

读者服务热线：(010)81055410　印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京崇工商广字第 0021 号

内容提要

R 具有强大的统计计算功能和便捷的数据可视化系统。本书重点讲解 R 的绘图系统，指导读者通过绘图系统实现数据可视化。

书中提供了快速绘制高质量图形的 150 多种技巧，每个技巧用来解决一个特定的绘图需求。读者可以通过目录快速定位到自己遇到的问题，查阅相应的解决方案。同时，作者在大部分的技巧之后会进行一些讨论和延伸，介绍一些总结出的绘图技巧。

本书侧重于解决具体问题，是 R 数据可视化的实战秘籍。本书中绝大多数的绘图案例都是以强大、灵活制图而著称的 R 包 ggplot2 实现的，充分展现了 ggplot2 生动、翔实的一面。从如何画点图、线图、柱状图，到如何添加注解、修改坐标轴和图例，再到分面的使用和颜色的选取等，本书都有清晰的讲解。虽然本书的大多数技巧使用的是 ggplot2，但是并不仅仅局限于 ggplot2 的介绍。作者的理念是用合适的工具来完成合适的绘图任务，读者也可以学到许多其他有用的绘图函数和工具，来适应各种复杂的需求。

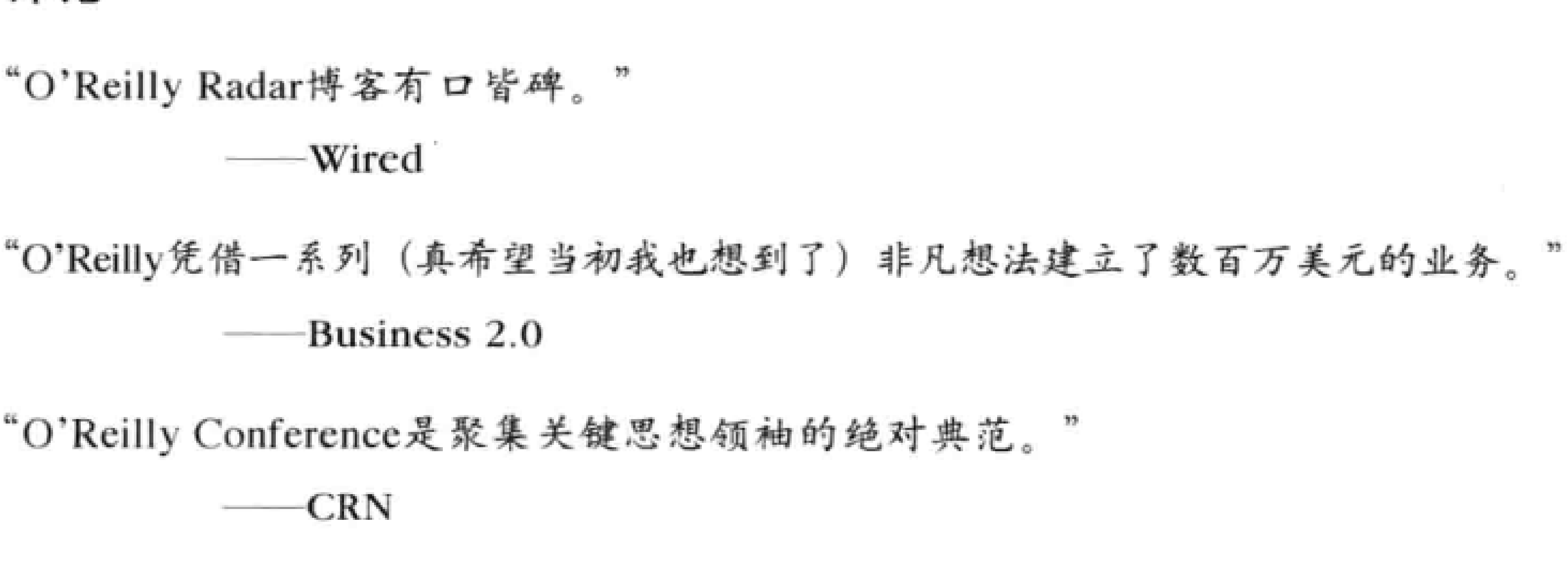
本书是学习 R 中丰富的数据可视化方法的权威手册，非常适合对 R 语言有基本的了解的读者阅读。

O'Reilly Media, Inc.介绍

O'Reilly Media通过图书、杂志、在线服务、调查研究和会议等方式传播创新知识。自1978年开始，O'Reilly一直都是前沿发展的见证者和推动者。超级极客们正在开创着未来，而我们关注真正重要的技术趋势——通过放大那些“细微的信号”来刺激社会对新科技的应用。作为技术社区中活跃的参与者，O'Reilly的发展充满了对创新的倡导、创造和发扬光大。

O'Reilly为软件开发人员带来革命性的“动物书”；创建第一个商业网站（GNN）；组织了影响深远的开放源代码峰会，以至于开源软件运动以此命名；创立了Make杂志，从而成为DIY革命的主要先锋；公司一如既往地通过多种形式缔结信息与人的纽带。O'Reilly的会议和峰会集聚了众多超级极客和高瞻远瞩的商业领袖，共同描绘出开创新产业的革命性思想。作为技术人士获取信息的选择，O'Reilly现在还将先锋专家的知识传递给普通的计算机用户。无论是通过书籍出版，在线服务或者面授课程，每一项O'Reilly的产品都反映了公司不可动摇的理念——信息是激发创新的力量。

业界评论



“O'Reilly Radar博客有口皆碑。”

——Wired

“O'Reilly凭借一系列（真希望当初我也想到了）非凡想法建立了数百万美元的业务。”

——Business 2.0

“O'Reilly Conference是聚集关键思想领袖的绝对典范。”

——CRN

“一本O'Reilly的书就代表一个有用、有前途、需要学习的主题。”

——Irish Times

“Tim是位特立独行的商人，他不光放眼于最长远、最广阔的视野并且切实地按照Yogi Berra的建议去做了：‘如果你在路上遇到岔路口，走小路（岔路）。’回顾过去Tim似乎每一次都选择了小路，而且有几次都是一闪即逝的机会，尽管大路也不错。”

——Linux Journal

译者序

R 官方网站的第一句话是这样介绍 R 语言的："R 是一个用于统计计算和绘图的自由软件环境。"这句话突出了 R 的两大特色：强大的统计计算功能和便捷的数据可视化系统。在很多情况下，当我们将 R 与其他同类的语言、软件进行比较时，通常都会强调其灵活的编程和计算能力，然而事实上，R 的绘图系统也是其最大的优势之一。

经过长年的开发和完善，目前 R 主要支持了四套图形系统：基础图形（base）、网格图形（grid）、lattice 图形和 ggplot2。其中前三个都内置于 R 的发行包，其功能已经非常稳定，而最为年轻的 ggplot2（最早开发于 2005 年）在历经若干次重大更新后，也逐渐成为了 R 中数据可视化的主流选择。由于 ggplot2 具有强大的语法特性和优雅的图形外观，它迅速吸引了众多的使用者和开发者，并已经开始被移植到其他语言中，如 Python、Julia 等。可以说，ggplot2 与 R 中传统的图形系统形成了良好的互补：后者适合快速的数据探索性分析，而前者则可以便捷地生成复杂的、高质量的统计图形。

本书中绝大多数的绘图案例都是基于 ggplot2 实现的，在某种意义上，本书可以认为是一部优秀的 ggplot2 入门和进阶手册。正如之前所说，ggplot2 之所以强大，是因为它不仅仅是一些函数的堆砌，而是有其内在的语法支持。因为这些特点，ggplot2 看上去更像是一门新的"语言"，从而需要使用者有一个学习和熟练的过程。

我们知道，学习一门语言（中文、英语、C++、R 等），大体都要经过一个从语法到词汇，再到句法应用的过程。ggplot2 的作者 Hadley Wickham 曾写过一本介绍 ggplot2 核心思想的书籍《ggplot2: Elegant Graphics for Data Analysis》，其中介绍了 ggplot2 的基本概念和用法，相当于为读者搭起了 ggplot2 的"语法"框架，而本书则是一本丰富、有趣的"词汇和例句书"。换句话说，本书侧重于解决具体问题，是一本实战秘籍：通过它读者可以了解到 ggplot2 生动、翔实的一面。比如，从如何画点图、线图、柱状图，到如何添加注解、修改坐标轴和图例，再到分面的使用和颜色的选取等，本书都有清晰的讲解。

本书由 150 多个精选的"技巧"组成，每个"技巧"都用来解决一个特定的绘图需求。读者可以通过章节的目录快速定位到自己遇到的问题，然后查阅相应的解决方案。同时，作者在大部分的"技巧"之后会进行一些讨论和延伸，介绍一些总结出的绘图技巧。就我自己的经历而言，在阅读本书之前，有时用 ggplot2 作图会遇到一些非常细节的问题，例如，如何调整条形图的顺序，如何修改图例的外观，如何选择文字的大

小和字体？这些问题都会让我花费很多时间来搜索解决方案。而在我把这本书通读一遍之后，很多问题立刻迎刃而解。我相信，本书对于 ggplot2 和数据可视化的学习都是大有裨益的。

当然，本书并不仅仅局限于 ggplot2 的介绍。作者的理念是用合适的工具来完成合适的绘图任务，所以在本书的第 2 章、第 13 章和第 14 章，读者也可以学到许多其他有用的绘图函数和工具，来适应各种复杂的需求。

本书的三位译者为本书中文版的面世付出了大量的时间和精力，其中肖楠翻译了第 7~11 章和附录，邓一硕翻译了第 2~6 章，魏太云翻译了前言、第 1 章和第 12~15 章。三位译者均是统计之都（http://cos.name/）的成员，他们也是当前国内 R 语言社区的领军人物，曾翻译了许多 R 语言的经典教材和书籍，并通过组织中国 R 语言会议、开办 COS 数据分析沙龙、参与论坛问题讨论、撰写博客、整理文档、编写软件包等多种方式为 R 语言社区做出了诸多贡献。值得一提的是，本书 13.1 节中介绍的 corrplot 软件包，正是由译者之一的魏太云开发完成的。统计之都团队作为国内 R 语言最早的一批布道者之一，有幸见证了 R 语言在国内兴起的整个过程。从网站创始人谢益辉开始在个人博客上连载 R 语言的教程和心得，到 2008 年第一届中国 R 语言会议的召开，再到如今众多 R 语言中文版书籍的面世，我相信其中的每一步都是国内统计学、以及更广义的数据科学不断向前发展的印记。我们也希望更多的有志之士加入到统计之都的团队之中，为国内统计学和数据科学贡献自己的力量。

本书的译者力求使翻译准确、生动，但疏漏之处在所难免，欢迎读者予以指正。为了让读者方便地获取和提交勘误信息，我们建立了本书的翻译项目页面：https://github.com/cosname/gcookbook-translation。读者也可以在统计之都的图书出版栏目（http://cos.name/books/）留言或提问。

本书的出版离不开众多人士的大力帮助，我们要郑重感谢爱荷华州立大学的王芯同学、中国人民大学的陈森同学和浙江大学的张政同学，他们在本书的翻译过程中提出了很多中肯的意见。没有他们的帮助，本书很难完成。此外，我们还要感谢人民邮电出版的杨海玲女士和编辑们，他们的专业精神让我叹服。

邱怡轩

2013 年 12 月于普渡大学

译者介绍

肖楠，中南大学数学与统计学院统计学系在读博士，统计之都论坛R语言版版主。合作翻译出版了《R语言实战》、《ggplot2：数据分析与图形艺术》等图书，编写了protr、Rcpi等R软件包。关注领域为统计机器学习、化学信息学与生物信息学、定量与系统药理学。

邓一硕，毕业于中央财经大学统计与数学学院，统计之都论坛金融投资分析版版主，现效力于首钢总公司计财部。擅长的领域为时间序列分析以及数据挖掘在金融投资分析中的应用。

魏太云，毕业于中国人民大学统计学院，统计之都理事会主席。合作翻译出版了《ggplot2：数据分析与图形艺术》等图书，参与编写了corrplot、recharts、knitr以及fun等R软件包。感兴趣的主题包括统计建模、机器学习和数据可视化。

审校者介绍

邱怡轩，普渡大学统计系在读博士，统计之都理事会成员。合作翻译出版了《ggplot2：数据分析与图形艺术》、《R语言编程艺术》等图书，参与编写了R2SWF、rARPACK、showtext、Layer、rationalfun、fun等R软件包。感兴趣的方向有函数型数据分析、统计计算和数据可视化等。

前言

几年前读研时我开始用 R，主要用来分析我在科研工作中收集到的数据。我使用 R 首先是想摆脱 SPSS 这样的统计软件的禁锢，即严格的环境和死板的分析。更何况，R 是免费的，所以我用不着说服别人为我购买一套这样的软件——这对一个穷研究生来说是相当的重要！此后，随着我对 R 的了解不断深入，我才发现原来 R 还可以绘制出非常优秀、动人的数据图形。

本书的每个“技巧”中，都列出了一个问题和对应的解决方法。在大多数情况下，我提供的并不是 R 中唯一的实现方法，但却是我认为的最佳方案。R 如此受欢迎的一个重要原因是它有很多附加的软件包，每一个软件包都为 R 提供了一些独特的功能。在 R 中也有很多可视化方面的软件包，但本书主要使用 ggplot2（声明：我现在工作的一部分就是开发 ggplot2；但是，在我还没意识到我可能会从事与 ggplot2 相关的工作之前，我已经完成了本书的大部分工作）。

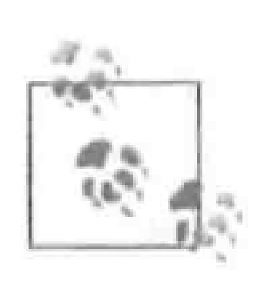
本书并不想罗列五花八门的方法，成为 R 数据可视化的综合手册；但是我希望当你想绘制所需图形的时候，本书能够对你有所帮助。或者说，当你不知道怎么画的时候，翻一翻这本书或许就可以找到一些可行的方案。

方法

本书面向的读者需要对 R 至少有一些基本的了解。书中的技巧会让你明白如何解决一些特定的问题。在使用例子的时候，我力图简单明了，这样你就会明白它们的工作机理，并可以方便地把解决方法应用到自己的问题上。

软件和平台说明

书中的大部分“技巧”都是用 ggplot2 完成的，有些“技巧”需要 ggplot2 的最新版本 0.9.3，这样也就要求有一个版本相对较新的 R——你可以在 R 的官方网站获取最新版本的 R。



如果你对 ggplot2 不熟悉，请参阅附录 A，那里对该包有一个简要的说明。

安装了 R 后，你可以再安装一些必要的包。除了 ggplot2 之外，你还可以选择安装 gcookbook 包（它包含了本书大多数例子的数据集）。要同时安装这两个包，只需运行命令：

```
install.packages("ggplot2")
install.packages("gcookbook")
```

你可能会被问到选择 CRAN（Comprehensive R Archive Network）镜像的问题。一般而言，任何一个镜像都可以正常工作，不过最好选择一个离你更近的，因为这样速度会更快。安装完包后，每次需要使用 ggplot2 包时在 R 会话中运行：

```
library(ggplot2)
```

本书中的技巧总是假设你已经加载了 ggplot2，所以不会显示这一行代码。

如果你看到这样的错误，意味着你忘记了加载 ggplot2。

```
错误：找不到函数 "ggplot"
```

英文版 R 的错误提示是：

```
Error: could not find function "ggplot"
```

R 的主要运行平台是 Mac OS X、Linux 和 Windows，本书中所有的“技巧”都可以在这些平台上运行。在保存位图输出的时候，会有平台的差异，详情参见第 14 章。

本书的排版约定

本书采用的体例如下：

- 等宽字体（`Constant width`）：表示程序清单，以及段落中引用的程序元素，如变量或函数名、数据库、数据类型、环境变量、语句和关键字。
- 加粗的等宽字体（**`Constant width bold`**）：表示需要用户手动输入的命令或其他文本。
- 斜的等宽字体（*`Constant width italic`*）：表示应该用用户所提供的值或根据上下文确定的值来替换的文本。



这个图标表示一个提示、建议或者一般的注记。

代码示例的使用

本书的目标是帮助你完成工作。一般而言，你可以在自己的程序和文档中使用本书中

的代码，如果你要复制的不是很大一部分代码，则无须取得我们的许可。例如，你可以在程序中使用本书中的多个代码块，无须获取我们许可。但是，要销售或分发来源于 O'Reilly 图书中的示例的光盘则必须得到许可。通过引用本书中的示例代码来回答问题时，不需要事先获得我们的许可。但是，如果你的产品文档中融合了本书中的大量示例代码，则需要取得我们的许可。

我们很欢迎引用时给出署名，但不做要求。署名一般包括书名、作者、出版社和 ISBN。例如 "R Graphics Cookbook by Winston Chang (O'Reilly). Copyright 2013 Winston Chang, 978-1-449-31695-2"。

如果你觉得你对例子代码的使用已经超出了合理程度或者上文所述的许可范围，请通过 permissions@oreilly.com 联系我们。

我们的联系方式

如果你想就本书发表评论或有任何疑问，敬请联系出版社。

美国：

O'Reilly Media Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472

中国：

北京市西城区西直门南大街 2 号成铭大厦 C 座 807 室（100035）
奥莱利技术咨询（北京）有限公司

我们还为本书建立了一个网页，其中包含了勘误表、示例和其他额外的信息。你可以通过如下地址访问该网页：

http://oreil.ly/R_Graphics_Cookbook

关于本书的技术性问题或建议，请发邮件到：

bookquestions@oreilly.com

欢迎登录我们的网站（http://www.oreilly.com），查看更多我们的书籍、课程、会议和最新动态等信息。

我们的其他联系方式如下：

Facebook: http://facebook.com/oreilly

Twitter: http://twitter.com/oreillymedia

YouTube: http://www.youtube.com/oreillymedia

致谢

没有一本书的诞生可完全归结为个人的成果。本书的完成得到了很多人直接或间接的帮助。我要感谢 R 社区创造并培育了一个积极活跃的生态系统。非常感谢 Hadley Wickham 的诸多帮助：他编写了本书所依赖的软件包 ggplot2，并在 O'Reilly 出版社考虑出版 R 图形书籍的时候推荐了我，他还为我打开了一扇深入了解 R 的窗户。

感谢本书的技术审稿人 Paul Teetor、Hadley Wickham、Dennis Murphy 和 Erik Iverson。他们渊博的知识和对细节的重视极大地提高了本书的质量。我还要感谢 O'Reilly 出版社积极推进此书的编辑们：Mike Loukides 在初始阶段给了我很多指导，Courtney Nash 伴我走到了最后。此外，我还要郑重感谢 O'Reilly 的 Holly Bauer 及制作团队的其他成员，他们从头到尾耐心地做了很多细致的编辑工作，并为此书增添了不少特色。

最后，我还要感谢我的妻子 Sylia，感谢她一贯的支持和理解——当然不只是在写本书的时候。

目录

第1章

R基础

本章包括以下基础知识：安装包、使用包和加载数据。

如果你想快速上手，本书大多数技巧都需要安装 ggplot2 和 gcookbook 包。运行下面命令来安装：

```
install.packages(c("ggplot2", "gcookbook"))
```

然后，在每个 R 会话中，你需要在运行本书的例子之前先加载它们：

```
library(ggplot2)
library(gcookbook)
```



附录 A 提供了一个关于 ggplot2 绘图包的简介，主要是面向不熟悉 ggplot2 的读者。

R 中的包是一些为了便于分发和传播而封装在一起的函数和（或）数据集（可以没有数据集）的集合。在你的电脑中安装软件包，便可以扩展 R 的功能。如果一个 R 用户编写了一个包并觉得这个包对其他 R 用户可能有用，那么，这位 R 用户就可以通过软件包仓库将该包发布。发布 R 软件包的最主要的软件包仓库是 CRAN（Comprehensive R Archive Network），不过也有其他的仓库，如 Bioconductor 和 Omegahat。

1.1 安装包

问题

如何从 CRAN 安装 R 包？

方法

使用 `install.packages()` 函数来安装包，括号中写上要安装的包名。以安装 ggplot2

包为例，运行：

```
install.packages("ggplot2")
```

此时系统可能提示你选择一个下载镜像，可以选择离你最近的一个；如果想要确保包的版本是最新的，那就选择 Austria 站点，因为这是 CRAN 的主服务器。

讨论

当 R 安装一个包的时候，该包依赖的所有包也都会被自动安装。

CRAN 是 R 包的仓库，在全球范围内有很多镜像，它是 R 默认使用的库。此外，还有几个软件包仓库，如 Bioconductor，它是与基因组数据分析相关的包的软件包仓库。

1.2　加载包

问题

如何加载一个已经安装了的包？

方法

使用 `library()` 函数，括号中写上要加载的包名。以加载 ggplot2 包为例，运行：

```
library(ggplot2)
```

当然，必须确保要加载的包已经被安装了。

讨论

本书的大多数技巧都需要在运行代码前加载包，无论是为了绘图（ggplot2 包）还是为了加载例子中的数据集（MASS 和 gcookbook 包）。

R 的一个不寻常之处是软件包（package）和软件库（library）的术语区别。 尽管我们使用 `library()` 函数来加载包（package），但一个包并不是一个软件库；如果你不幸犯此错误，可能会激怒一些资深的 R 用户。

软件库指的是一个包含了若干软件包的目录。你既可以拥有一个系统级别的软件库，也可以针对每个用户单独设立一个软件库。

1.3　加载分隔符式的文本文件

问题

如何加载一个分隔符式的文本文件中的数据？

方法

加载逗号分隔组（CSV）数据的最常用方法是：

```
data <-read.csv("datafile.csv")
```

讨论

由于数据文件有许多不同的格式，为了加载它们，提供了很多对应的选项。如果一个数据集首行没有列名：

```
data <-read.csv("datafile.csv", header=FALSE)
```

得到的数据框的列名将是 V1、V2 等，你可能想要重命名列：

```
# 手动为列名赋值
names(data) <-c("Column1","Column2","Column3")
```

还可以用 sep 参数来设置分隔符号。如果是空格分隔，使用 sep=" "；如果是制表符分隔，使用 \t。

```
data <-read.csv("datafile.csv", sep="\t")
```

默认情况下，数据集中的字符串（string）会被视为因子（factor）处理。假设下面是你的数据文件，然后，你用 read.csv() 来读取：

```
"First","Last","Sex","Number"
"Currer","Bell","F",2
"Dr.","Seuss","M",49
"","Student",NA,21
```

得到的数据框将会把 First、Last 等存储为因子，尽管此时将它们视为字符串（或使用 R 中的术语，字符：character）更为合理。为了区别这一点，可以设置 stringsAsFactors=FALSE。如果有些列应该被处理为因子格式，你可以再逐个转换：

```
data <-read.csv("datafile.csv", stringsAsFactors=FALSE)

# 转换为因子
data$Sex <-factor(data$Sex)

str(data)

'data.frame': 3 obs. of 4 variables:
$ First : chr "Currer" "Dr." ""
$ Last : chr "Bell" "Seuss" "Student"
$ Sex : Factor w/ 2 levels "F","M": 1 2 NA
$ Number: int 2 49 21
```

或者，你可以在加载的时候不做设置（字符串自动转换为因子），加载之后再对需要的列进行因子到字符的转换。

另见

read.csv() 是对 read.table() 一个便捷的封装函数。如果需要更多的输入控制，参

见 ?read.table。

1.4 从 Excel 文件中加载数据

问题

如何从 Excel 文件中加载数据？

方法

xlsx 包中的函数 read.xlsx() 可以读取 Excel 文件，下面的代码将会读取 Excel 中的第一个工作表：

```
# 只需要安装一次
install.packages("xlsx")

library(xslx)
data <-read.xlsx("datafile.xlsx", 1)
```

如果需要阅读老版本的 Excel 文件（.xls 格式），gdata 包提供了函数 read.xls()：

```
# 只需要安装一次
install.packages("gdata")

library(gdata)
# 读取第一张工作表
data <-read.xls("datafile.xls")
```

讨论

使用 read.xlsx() 加载工作表时，既可以用序数参数 sheetIndex 来指定，也可以用工作表名参数 sheetName 来指定：

```
data <-read.xlsx("datafile.xls", sheetIndex=2)

data <-read.xlsx("datafile.xls", sheetName="Revenues")
```

使用 read.xls() 加载工作表时，可以用序数参数 sheet 来指定：

```
data <-read.xls("datafile.xls", sheet=2)
```

安装 xlsx 和 gdata 包时需要在电脑上安装其他软件。对于 xlsx 包，需要安装 Java；对于 gdata 包，需要安装 Perl。Perl 在 Linux 和 Mac OS X 上是系统自带的，但在 Windows 上没有。如果是在 Windows 上，你需要安装 ActiveState Perl，其社区版本可以免费获得（http: //www. active state.com/active perl）。

如果你不想这样折腾，更简单的替代方案是打开 Excel 文件后另存为标准的文本格式，比如 CSV。

另见

输入 ?read.xls 和 ?read.xlsx 来查看更多关于读取文件的选项。

1.5 从 SPSS 文件中加载数据

问题

如何从 SPSS 文件加载数据？

方法

foreign 包中的函数 read.spss() 可以读取 SPSS 文件。若要读取 SPSS 文件中的第一张表：

```
# 只需首次使用时安装
install.packages("foreign")

library(foreign)
data <-read.spss("datafile.sav")
```

讨论

foreign 包中还有很多读取其他格式文件的函数，包括以下几种。

- read.octave()：Octave 和 MATLAB。
- read.systat()：SYSTAT。
- read.xport()：SAS XPORT。
- read.dta()：Stata。

另见

输入 ls("package:foreign") 可以查看该包中的所有函数的列表。

第2章

快速探索数据

虽然本书中大部分图形都是通过 ggplot2 包绘制的，但这并不是 R 绘制图形的唯一方法。要快速探索数据，有时使用 R 基础包中的绘图函数会很有用。这些函数随 R 软件默认安装，无需另行安装附加包。它们简短易输入，处理简单问题时使用方便，且运行速度极快。

如果你想绘制较为复杂的图形，那么，转用 ggplot2 包通常是更好的选择。部分原因在于 ggplot2 提供了一个统一的接口和若干选项来替代基础绘图系统中对图形的修修补补和各种特例。一旦掌握了 ggplot2 的工作机制，你就可以应用这些知识来绘制从散点图、直方图到小提琴图和地图等各种统计图形了。

本章介绍的技巧演示了用基础绘图系统绘制统计图形的方法，也对如何用 `ggplot2` 中的 `qplot()` 函数绘制同样的图形做出了说明。`qplot()` 函数的语法与基础绘图系统类似，对于每一个由 `qplot()` 函数绘制的图形，技巧中也提供了用更强大的 `ggplot()` 函数来绘图的等价解决方案。

如果你已经知道如何使用基础图形系统，那么当你想绘制更复杂的图形时，可以将这些例子放在一起进行对比以帮助你过渡到 ggplot2 系统。

2.1 绘制散点图

问题

如何绘制散点图？

方法

使用 `plot()` 函数可绘制散点图（见图 2-1），运行命令时依次传递给 `plot()` 函数一个向量 ***x*** 和一个向量 ***y***。

```
plot(mtcars$wt,mtcars$mpg)
```

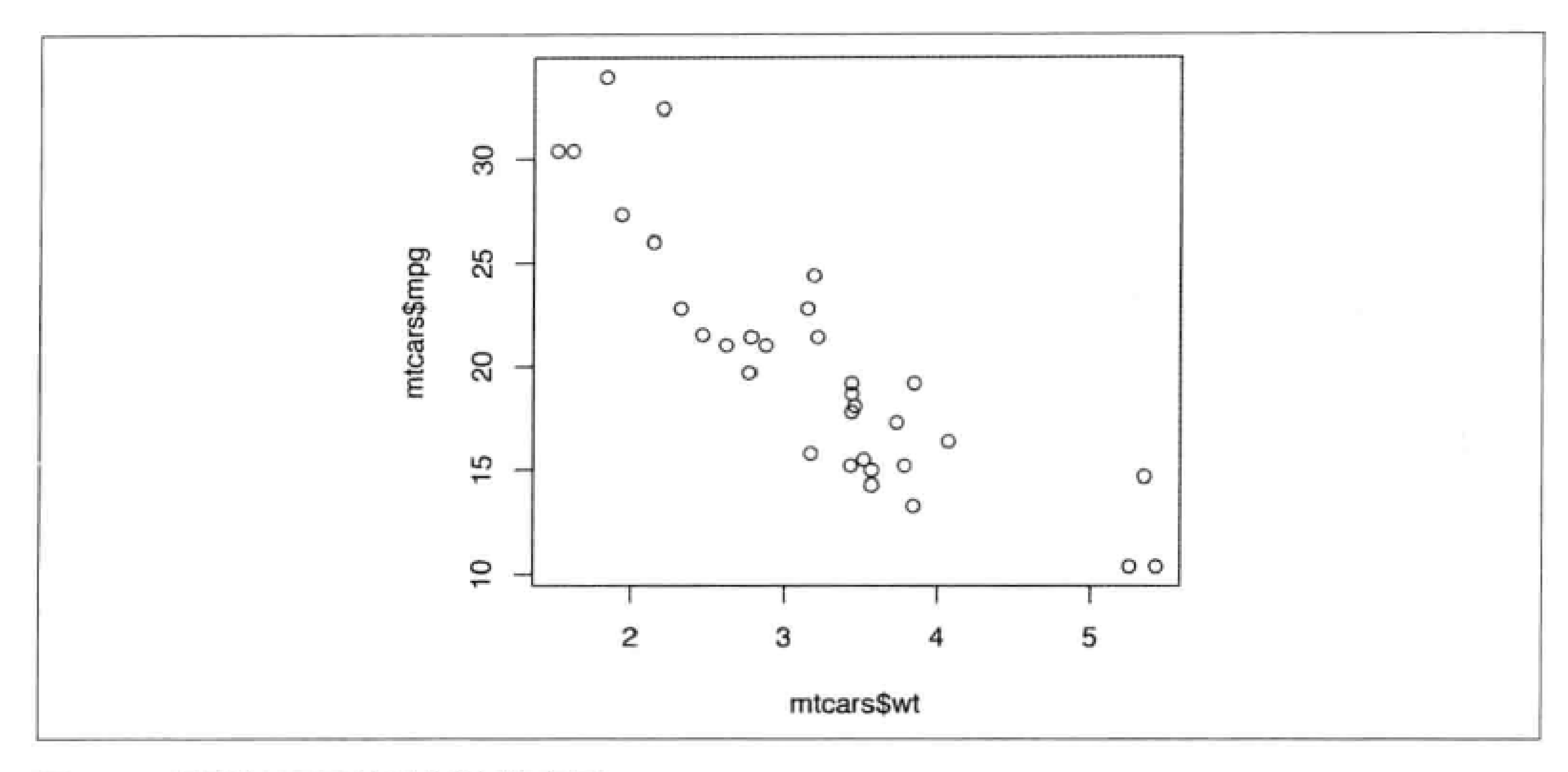


图 2-1　基础绘图系统绘制的散点图

对于 ggplot2 系统，可用 qplot() 函数得到相同的绘图结果（见图 2-2）：

```
library(ggplot2)
qplot(mtcars$wt, mtcars$mpg)
```

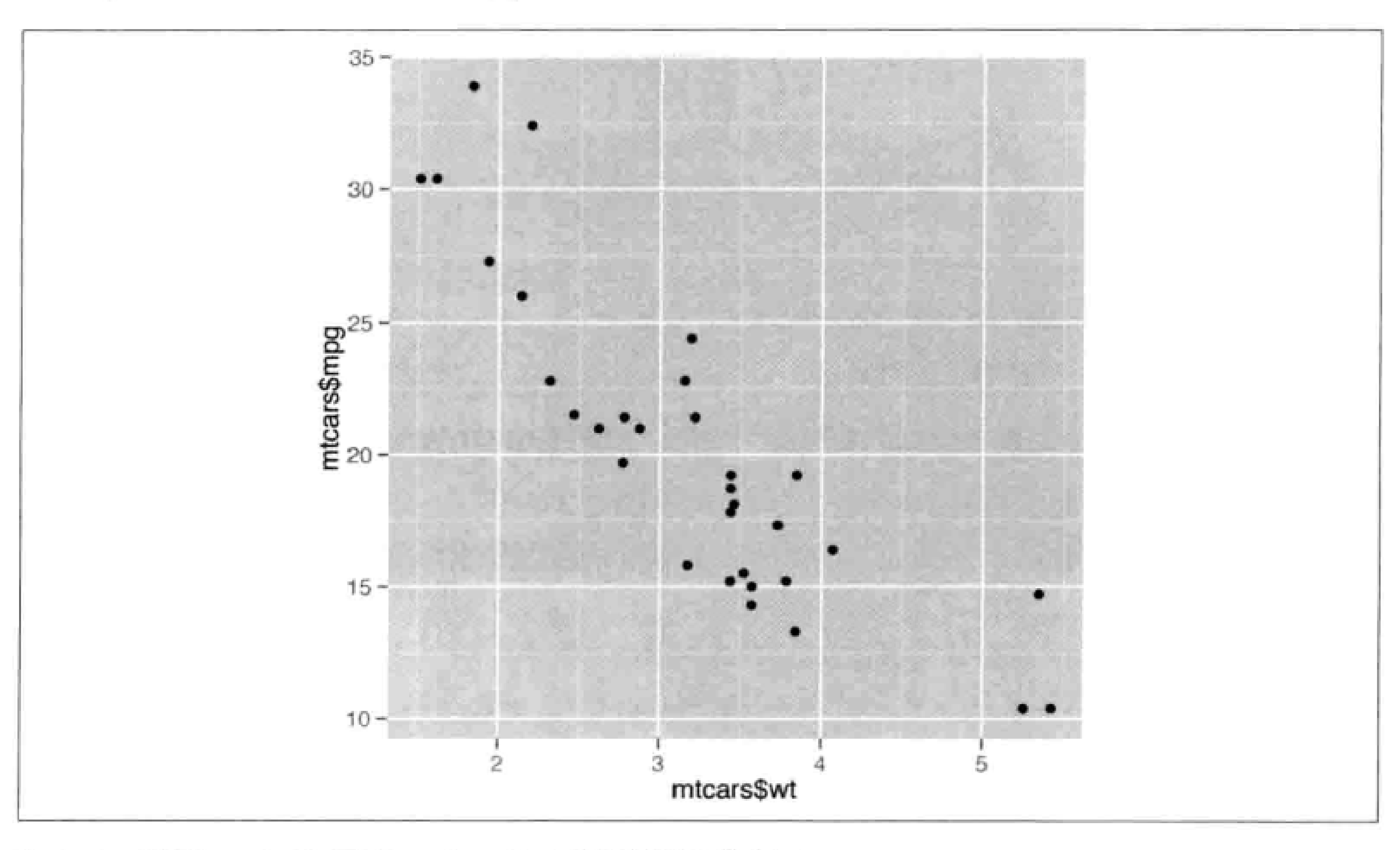


图 2-2　使用 ggplot2 包中 qplot() 函数绘制的散点图

如果绘图所用的两个参数向量包含在同一个数据框内，则可以运行下面的命令：

```
qplot(wt,mpg,data=mtcars)
# 这与下面等价
ggplot(mtcars, aes(x=wt, y=mpg)) + geom_point()
```

另见

更多关于绘制散点图的详细内容可参见本书第 5 章。

2.2 绘制折线图

问题

如何绘制折线图？

方法

使用 plot() 函数绘制折线图（见图 2-3 左图）时需向其传递一个包含 x 值的向量和一个包含 y 值的向量，并使用参数 type="l"：

```
plot(pressure$temperature, pressure$pressure, type="l")
```

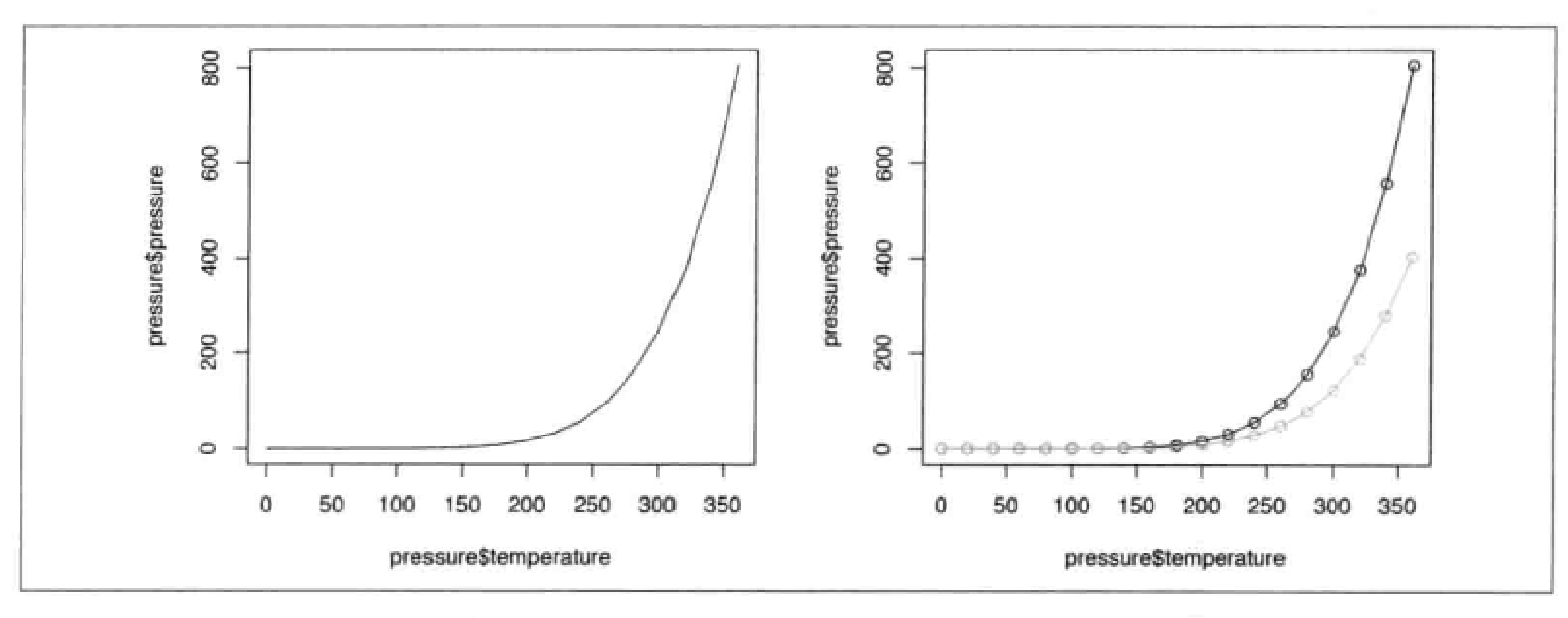


图 2-3　左图：基础绘图系统绘制的折线图　右图：向图形中添加数据点和另一条折线

如果要向图形中添加数据点或者多条折线（见图 2-3 右图），则需先用 plot() 函数绘制第一条折线，再通过 points() 函数和 lines() 函数分别添加数据点和更多折线：

```
plot(pressure$temperature, pressure$pressure, type="l")
points(pressure$temperature,pressure$pressure)

lines(pressure$temperature, pressure$pressure/2, col="red")
points(pressure$temperature, pressure$pressure/2, col="red")
```

在 ggplot2 中，可以使用 qplot() 函数并将参数设定为 geom="line" 得到类似的绘图结果（见图 2-4）：

```
library(ggplot2)
qplot(pressure$temperature, pressure$pressure, geom="line")
```

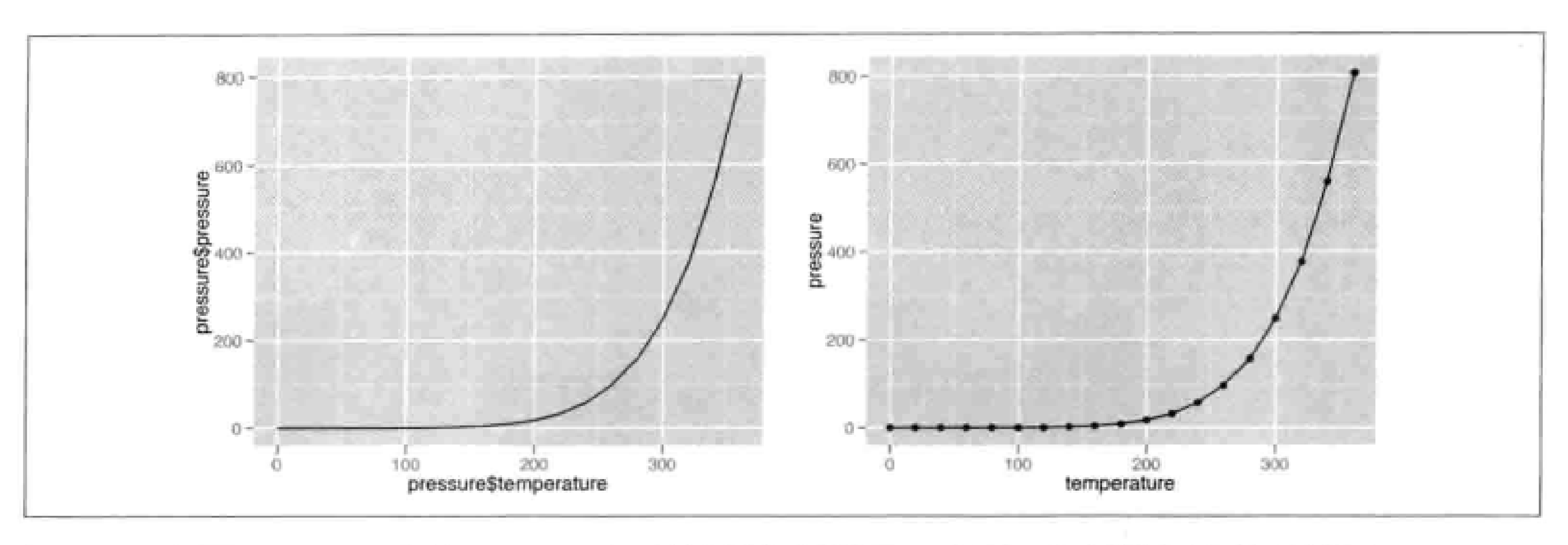


图 2-4　左图：ggplot2 中的 qplot() 函数绘制的折线图　右图：添加数据点的折线图

如果函数的两个参数向量已包含在同一个数据框中，则可以运行下面的语句：

```
qplot(temperature,pressure, data=pressure, geom="line")
# 这等价于下面的命令
ggplot(pressure, aes(x=temperature, y=pressure)) + geom_line()

# 添加数据点
qplot(temperature, pressure, data=pressure, geom=c("line", "point"))
# 这等价于下面的命令
ggplot(pressure, aes(x=temperature, y=pressure)) + geom_line() + geom_point()
```

另见

更多关于绘制折线图的详细内容可参见本书第 4 章。

2.3　绘制条形图

问题

如何绘制条形图？

方法

对变量的值绘制条形图（见图 2-5 左图），可以使用 barplot() 函数，并向其传递两个向量作为参数，第一个向量用来设定条形的高度，第二个向量用来设定每个条形对应的标签（可选）。

如果向量中的元素已被命名，则系统会自动使用元素的名字作为条形标签：

```
barplot(BOD$demand, names.arg=BOD$Time)
```

有时候，“条形图”表示的是分组数据中各个元素的频数（见图 2-5 右图）。这种条形图跟直方图有些类似，不过，其用离散取值的 x 轴替代了直方图中连续取值的 x 轴。要计算向量中各个类别的频数，可以使用 table() 函数。

```
table(mtcars$cyl)

 4  6  8
11  7 14
# 值为 4 的频数为 11,6 的为 7,8 的为 14
```

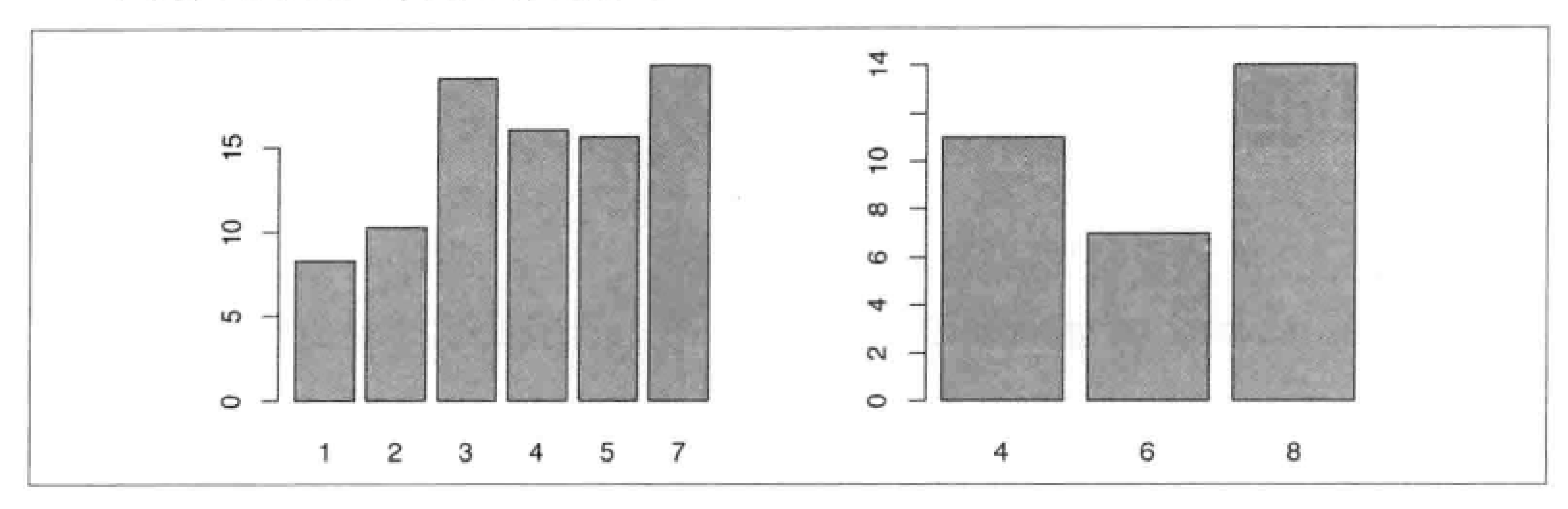


图 2-5　左图：基础绘图系统绘制的条形图　右图：向量元素的频数条形图

只需将上面的表格结果传递给 barplot() 函数即可绘制频数条形图：

```
# 生成频数表
barplot(table(mtcars$cyl))
```

对于 ggplot2 系统，可以使用 qplot() 函数得到类似的绘图结果（见图 2-6）。绘制变量值的条形图时需将参数设定为 geom="bar" 和 stat="identity"。注意变量 *x* 分别为连续取值和离散取值时输出结果的差异。

```
library(ggplot2)
qplot(BOD$Time, BOD$demand, geom="bar", stat="identity")
# 将 x 转化为因子型变量，令系统将其视作离散值
qplot(factor(BOD$Time), BOD$demand, geom="bar", stat="identity")
```

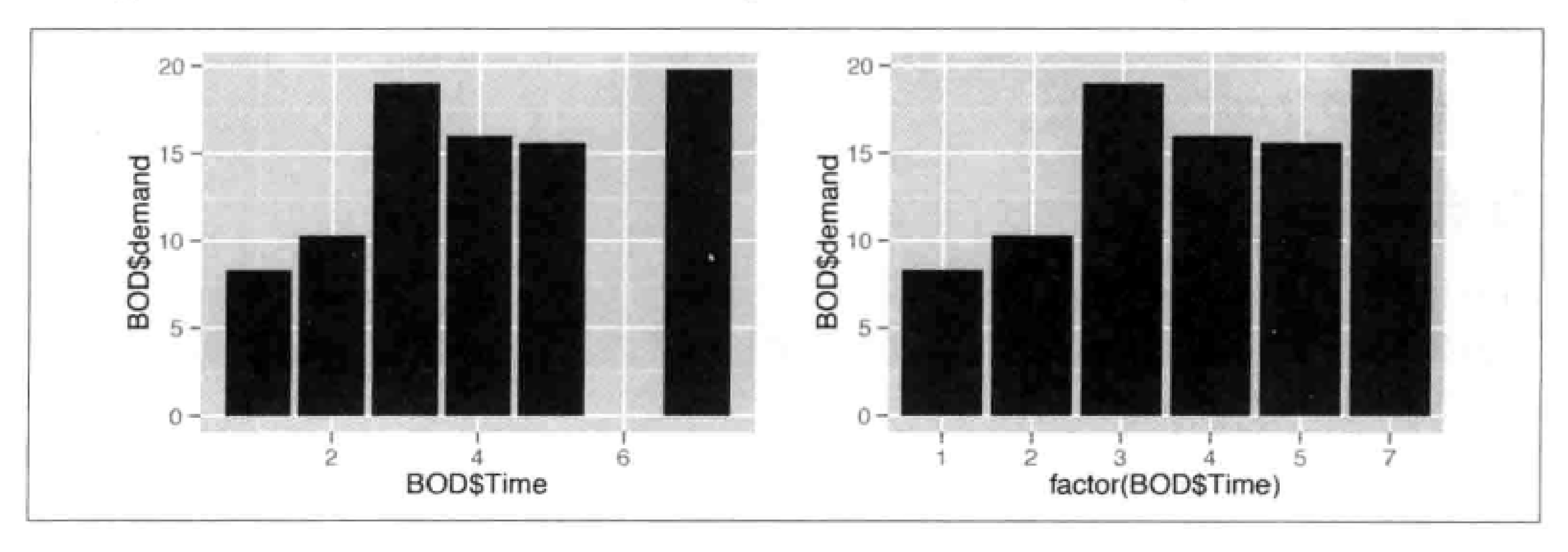


图 2-6　左图：qplot() 函数绘制的连续变量 x 的变量值条形图　右图：将变量 x 转化为因子型变量（注意，横坐标上没有 6 这个类别）

qplot() 函数也可以用来绘制分组变量的频数条形图（见图 2-7），事实上，这是 ggplot2 绘制条形图的默认方式，它比绘制变量值条形图的命令更简短。再提醒一次，

注意连续 *x* 轴和离散 *x* 轴的差异。

```
# cyl 是连续变量
qplot(mtcars$cyl)
# 将 cyl 转化为因子型变量
qplot(factor(mtcars$cyl))
```

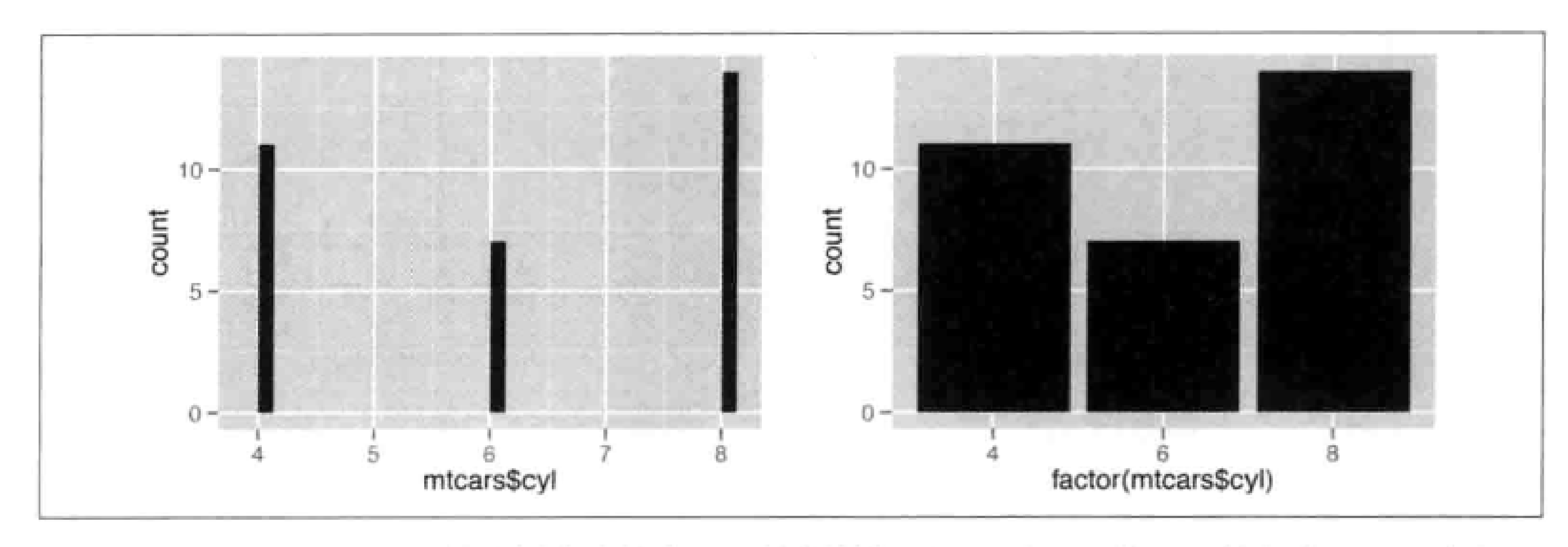


图 2-7　左图：`qplot()` 函数绘制的连续变量 x 的频数条形图　右图：将 `cyl` 转化为因子型变量

如果参数向量包含在同一个数据框内，则可以运行下面的语句：

```
# 变量值条形图，这里用 BOD 数据框中的 Time 列
# 和 demand 列分别作为 x 和 y 参数
qplot(Time, demand, data=BOD, geom="bar", stat="identity")
# 这与下面的语句等价
ggplot(BOD, aes(x=Time, y=demand)) + geom_bar(stat="identity")

# 频数条形图
qplot(factor(cyl), data=mtcars)
# 这与下面的语句等价
ggplot(mtcars, aes(x=factor(cyl))) + geom_bar()
```

另见

更多关于绘制条形图的详细内容可参见本书第 3 章。

2.4　绘制直方图

问题

如何绘制直方图来查看一维数据的分布特征？

方法

可以使用 `hist()` 函数绘制直方图（见图 2-8），使用时需向其传递一个向量：

```
hist(mtcars$mpg)

# 通过 breaks 参数指定大致组距
```

```
hist(mtcars$mpg,breaks=10)
```

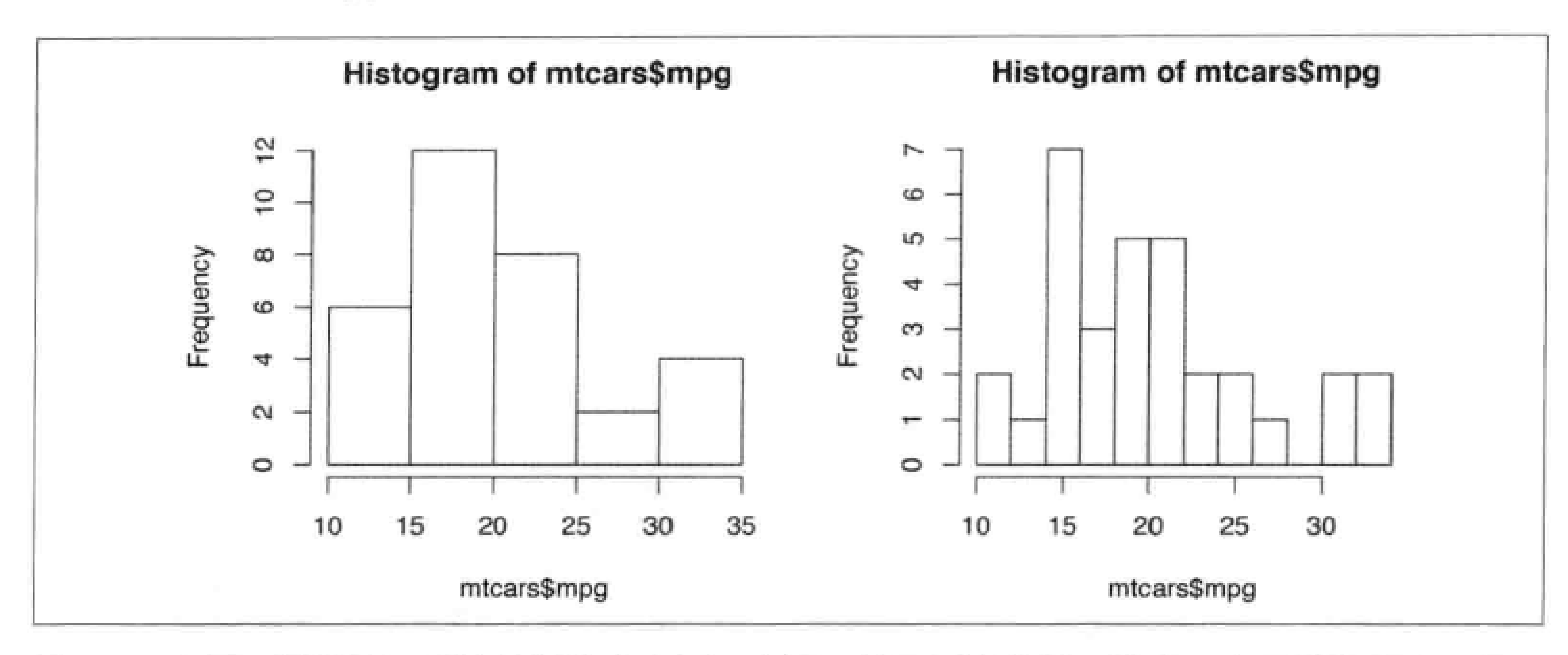


图 2-8　左图：基础绘图系统绘制的直方图　右图：使用更多分组。注意：由于组距变小，每组对应的样本数有所减少

对于 ggplot2 包，可以使用 qplot() 函数得到同样的绘图结果（见图 2-9）：

```
qplot(mtcars$mpg)
```

如果参数向量在同一个数据框内，则可以使用下面的语句：

```
library(ggplot2)
qplot(mpg, data=mtcars, binwidth=4)
# 这等价于
ggplot(mtcars, aes(x=mpg)) + geom_histogram(binwidth=4)
```

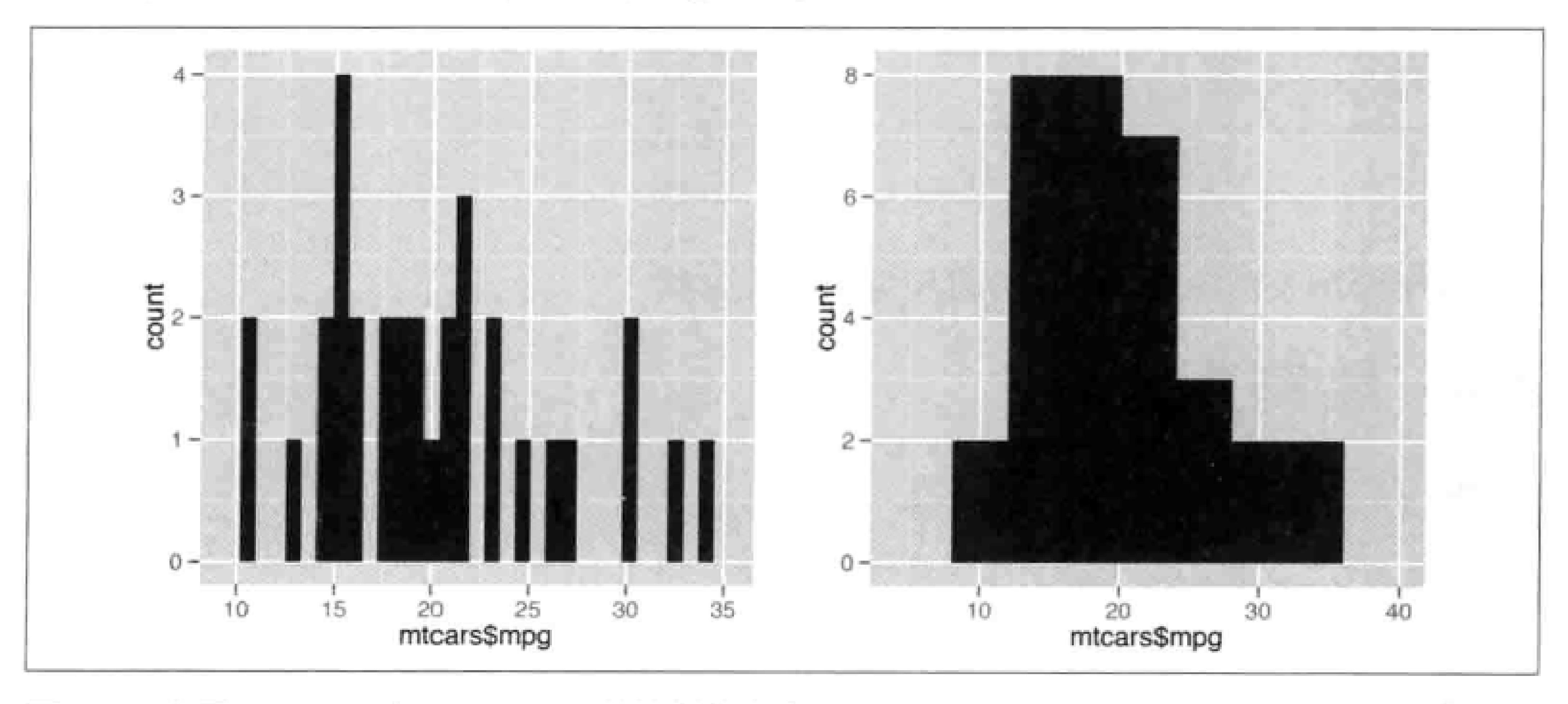


图 2-9　左图：ggplot2 中 qplot() 函数绘制的直方图，组距为默认值　右图：组距更大的直方图

另见

更多关于绘制直方图的内容参见 6.1 节和 6.2 节。

2.5 绘制箱线图

问题

如何绘制箱线图以对不同分布进行比较？

方法

使用 `plot()` 函数绘制箱线图（见图 2-10）时向其传递两个向量：***x*** 和 ***y***。当 ***x*** 为因子型变量（与数值型变量对应）时，它会默认绘制箱线图：

```
plot(ToothGrowth$supp, ToothGrowth$len)
```

当两个参数向量包含在同一个数据框中时，也可以使用公式语法。公式语法允许我们在 *x* 轴上使用变量组合，如图 2-10 所示。

```
# 公式语法
boxplot(len ~ supo, data = ToothGrowth)
# 在 x 轴上引入两变量的交互
boxplot(len ~ supp + dose, data = ToothGrowth)
```

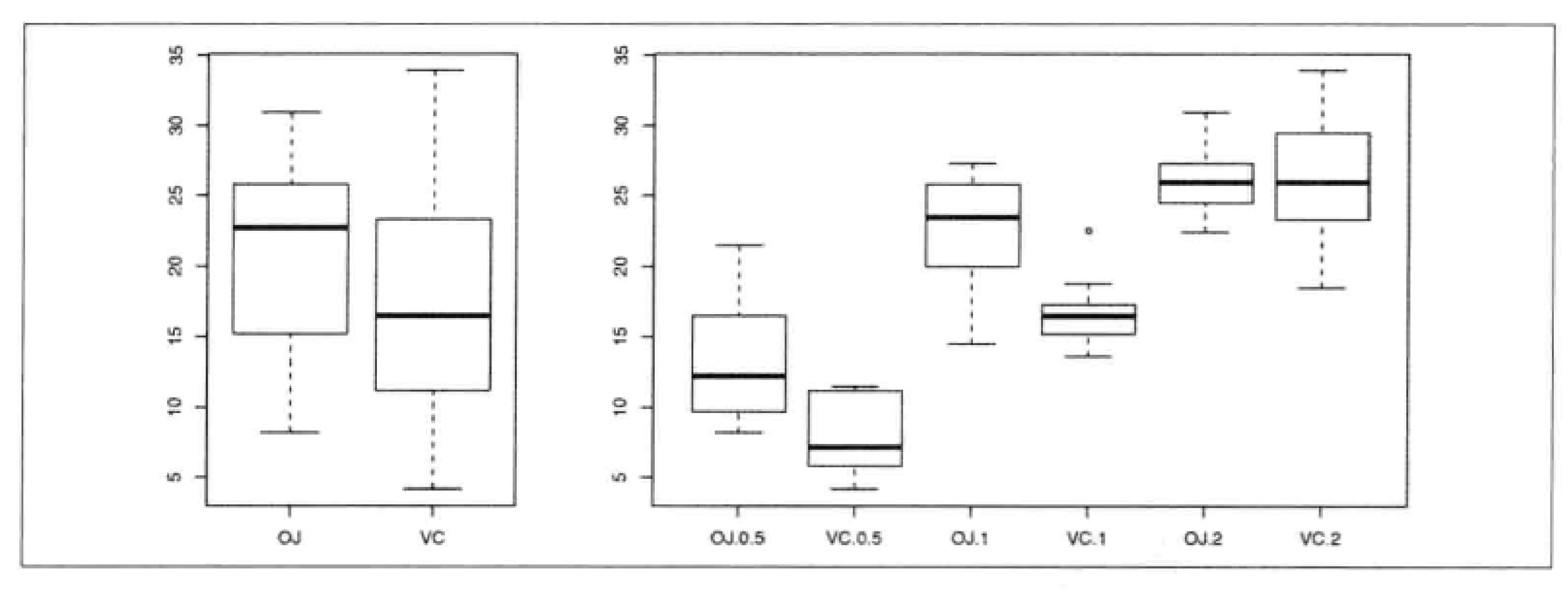


图 2-10　左图：基础绘图系统绘制的箱线图　右图：基于多分组变量的箱线图

对于 ggplot2 包，你可以使用 `qplot()` 函数绘制同样的图形（见图 2-11），使用时将参数设定为 `geom="boxplot"`：

```
library(ggplot2)
qplot(ToothGrowth$supp, ToothGrowth$len, geom="boxplot")
```

当两个参数向量在同一个数据框内时，则可以使用下面的语句：

```
qplot(supp, len, data=ToothGrowth, geom="boxplot")
# 这等价于
ggplot(ToothGrowth, aes(x=supp,y=len)) + geom_boxplot()
```

使用 `interaction()` 函数将分组变量组合在一起也可以绘制基于多分组变量的箱线图，如图 2-11 右图所示。本例中，`dose` 变量是数值型，因此，我们必须先将其转化为因

子型变量，再将其作为分组变量：

```
# 使用三个独立的向量参数
qplot(interaction(ToothGrowth$supp, ToothGrowth$dose), ToothGrowth$len,geom="boxplot")
# 也可以以数据框中的列作为参数
qplot(interaction(supp, dose), len, data=ToothGrowth, geom="boxplot")
# 这等价于
ggplot(ToothGrowth, aes(x=interaction(supp, dose), y=len)) + geom_boxplot()
```

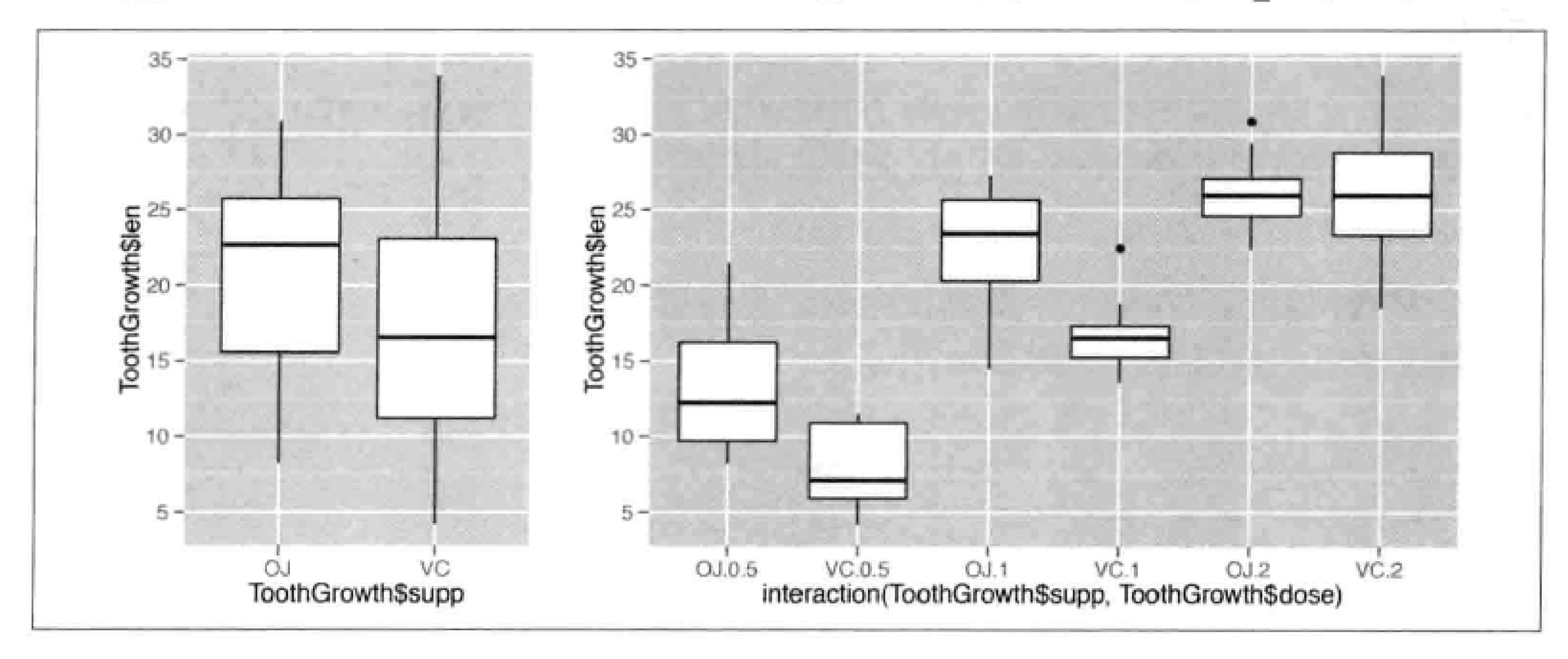
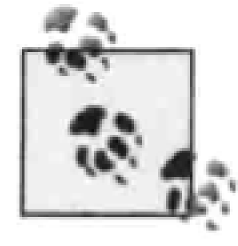


图 2-11 左图：qplot() 函数绘制的箱线图 右图：基于多分组变量的箱线图



你可能会注意到基础绘图系统绘制的箱线图与 ggplot2 略有不同。这是因为两者在绘图过程中调用的计算分位数的方法略有差异。运行 ?geom_boxplot 和 ?boxplot.base 命令可以得到更多关于两者差异的细节信息。

另见

更多关于绘制箱线图的内容参见 6.6 节。

2.6 绘制函数图像

问题

如何绘制函数图像？

方法

可以使用 curve() 函数绘制函数图像，如图 2-12 左图所示。使用时需向其传递一个关于变量 x 的表达式：

```
curve(x^3 - 5*x, from=-4, to=4)
```

你可以绘制任何一个以数值型向量作为输入且以数值型向量作为输出的函数图像，包括你自己定义的函数，如图 2-12 右图所示。

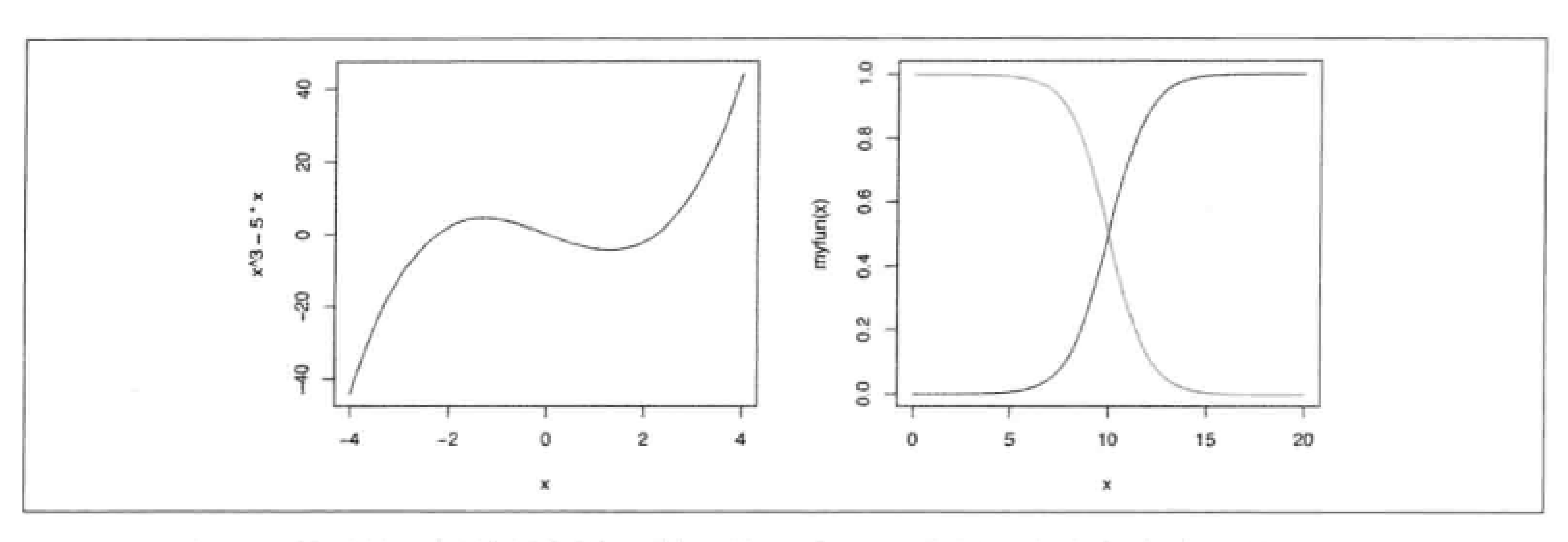


图 2-12　左图：基础绘图系统绘制的函数图像　右图：绘制用户自定义的函数

将参数设置为 add=TRUE 可以向已有图形添加函数图像：

```
# 绘制用户自定义的函数图像
myfun <- function(xvar) {
    1/(1 + exp(-xvar + 10))
}
curve(myfun(x), from=0, to=20)
# 添加直线
curve(1-myfun(x), add = TRUE, col ="red")
```

对于 ggplot2，可以使用 qplot() 函数绘制得到同样的结果（见图 2-13）。使用时需设定 stat="function" 和 geom="line"，并向其传递一个输入和输出皆为数值型向量的函数：

```
library(ggplot2)
# 将 x 轴的取值范围设定为 0 到 20
qplot(c(0, 20), fun=myfun, stat="function", geom="line")
# 这等价于
ggplot(data.frame(x=c(0, 20)), aes(x=x)) + stat_function(fun=myfun, geom="line")
```

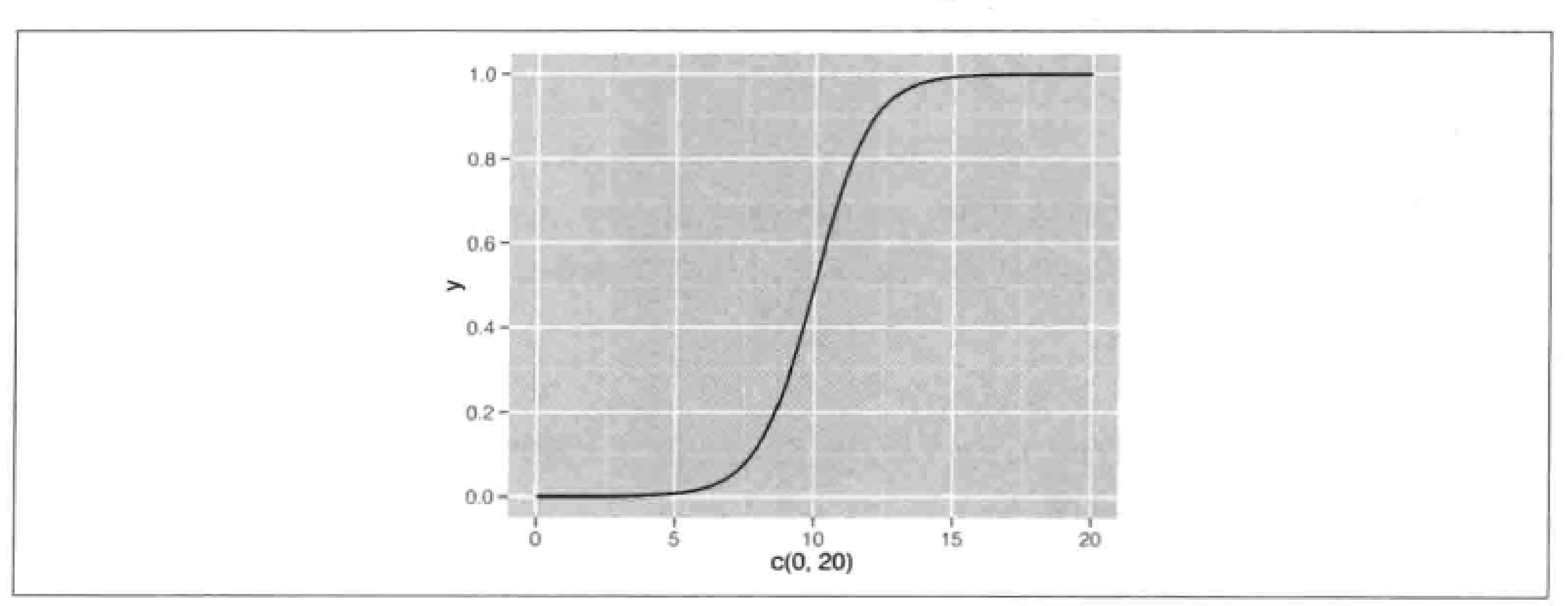


图 2-13　qplot() 函数绘制的函数图像

另见

更多关于绘制函数图像的内容参见 13.2 节。

第3章

条形图

条形图也许是最常用的数据可视化方法，通常用来展示不同的分类下（在 x 轴上）某个数值型变量的取值（在 y 轴上）。例如，条形图可以用来形象地展示四种不同商品的价格情况，但不适宜用来展示商品价格随时间的变动趋势，因为这里时间是一个连续变量——尽管我们也可以这么做，后面会看到这种情形。

绘制条形图时需特别注意一个重要的细节：有时条形图的条形高度表示的是数据集中变量的频数，有时则表示变量取值本身。牢记这个区别——这里极易混淆，因为两者与数据集的对应关系不同，但又对应同样的术语。本章将对此进行深入讨论，并分别介绍这两类条形图的绘图技巧。

3.1 绘制简单条形图

问题

你有一个包含了两列数据的数据框，其中一列数据表示条形在 x 轴上的位置，另一列表示每个条形在 y 轴上对应的高度，基于此，如何绘制条形图？

方法

使用 ggplot() 函数和 geom_bar(stat="identity") 绘制上述条形图，并分别指定与 x 轴和 y 轴对应的变量（见图 3-1）。

```
library(gcookbook) # 为了使用数据
ggplot(pg_mean, aes(x=group, y=weight)) + geom_bar(stat="identity")
```

讨论

当 x 是连续型（数值型）变量时，条形图的结果与上图会略有不同。此时，ggplot 不是只在实际取值处绘制条形，而将在 x 轴上介于最大值和最小值之间所有可能的取值处绘

制条形，如图 3-2 所示。我们可以使用 factor() 函数将连续型变量转化为离散型变量。

```
# 没有 Time == 6 的输入
BOD

 Time demand
    1    8.3
    2   10.3
    3   19.0
    4   16.0
    5   15.6
    7   19.8

# Time 是数值型（连续型）变量
str(BOD)

'data.frame':    6 obs. of  2 variables:
 $ Time  : num  1 2 3 4 5 7
 $ demand: num  8.3 10.3 19 16 15.6 19.8
 - attr(*, "reference")= chr "A1.4, p. 270"

ggplot(BOD, aes(x=Time, y=demand)) + geom_bar(stat="identity")

# 使用 factor() 函数将 Time 转化为离散型（分类）变量
ggplot(BOD, aes(x=factor(Time), y=demand)) + geom_bar(stat="identity")
```

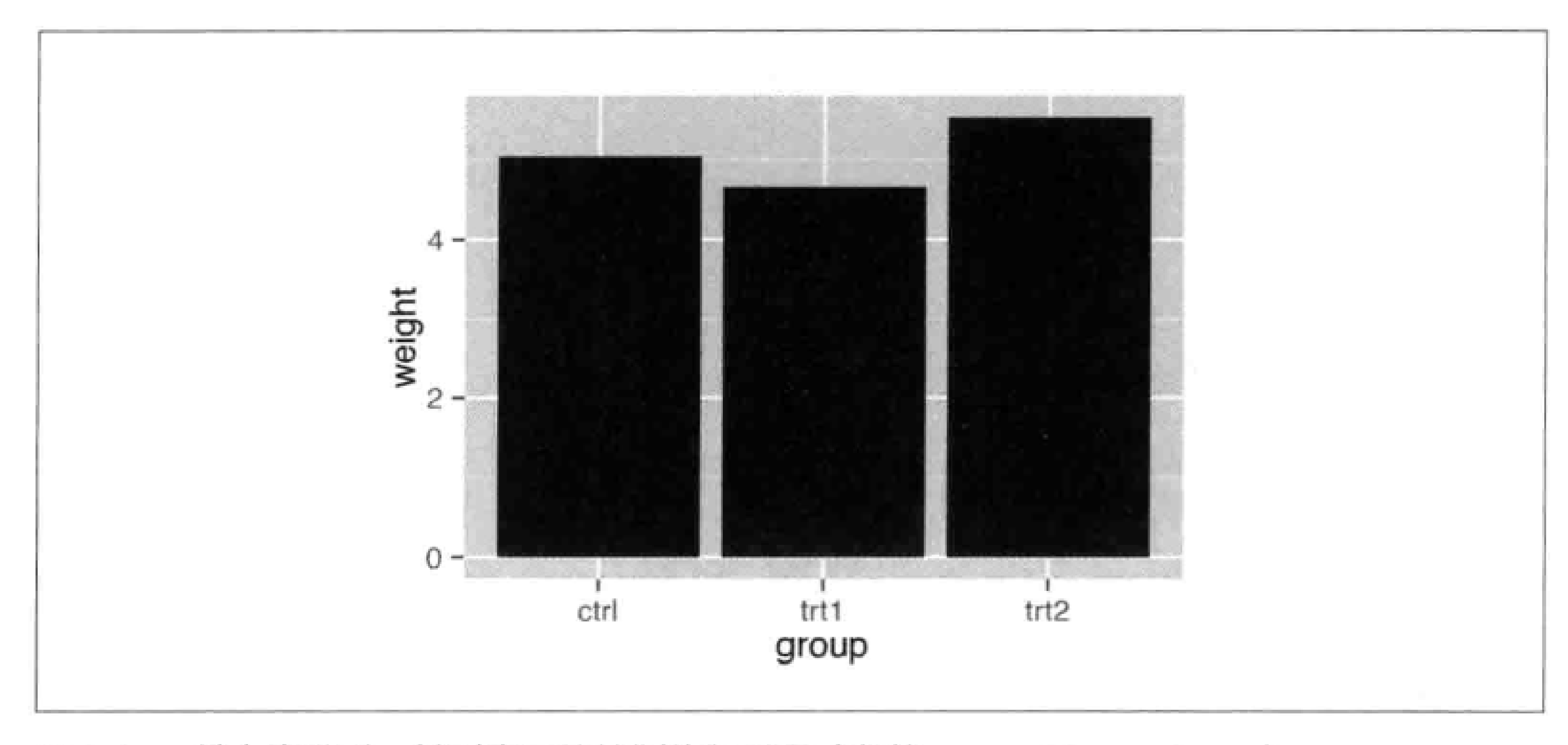


图 3-1　x 轴为离散时，针对变量值绘制的条形图（参数 stat="identity"）

本例中，数据集中包含两列分别对应于 x 和 y 变量。如果你想让条形图的高度与每组变量的频数相对应，可参见 3.3 节的内容。

默认设置下，条形图的填充色为黑灰色且条形图没有边框线，我们可通过调整 fill 参数的值来改变条形图的填充色；可通过 colour 参数为条形图添加边框线。在图 3-3 中，我们将填充色和边框线分别指定为浅蓝色和黑色。

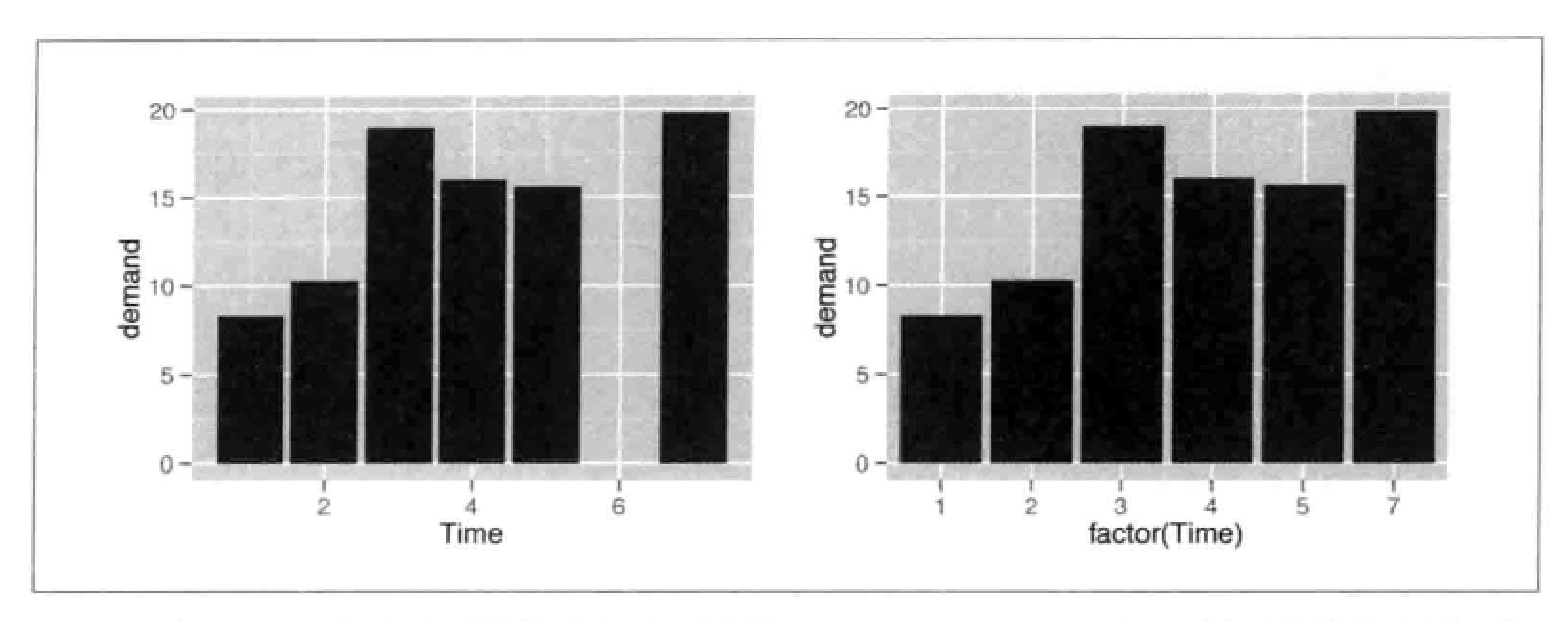


图 3-2　左图：针对变量值绘制的条形图（参数 stat="identity"），*x* 轴对应的是连续型变量　　右图：将 *x* 转化为因子型变量之后绘制的条形图（注意此处缺失了取值为 6 的条形）

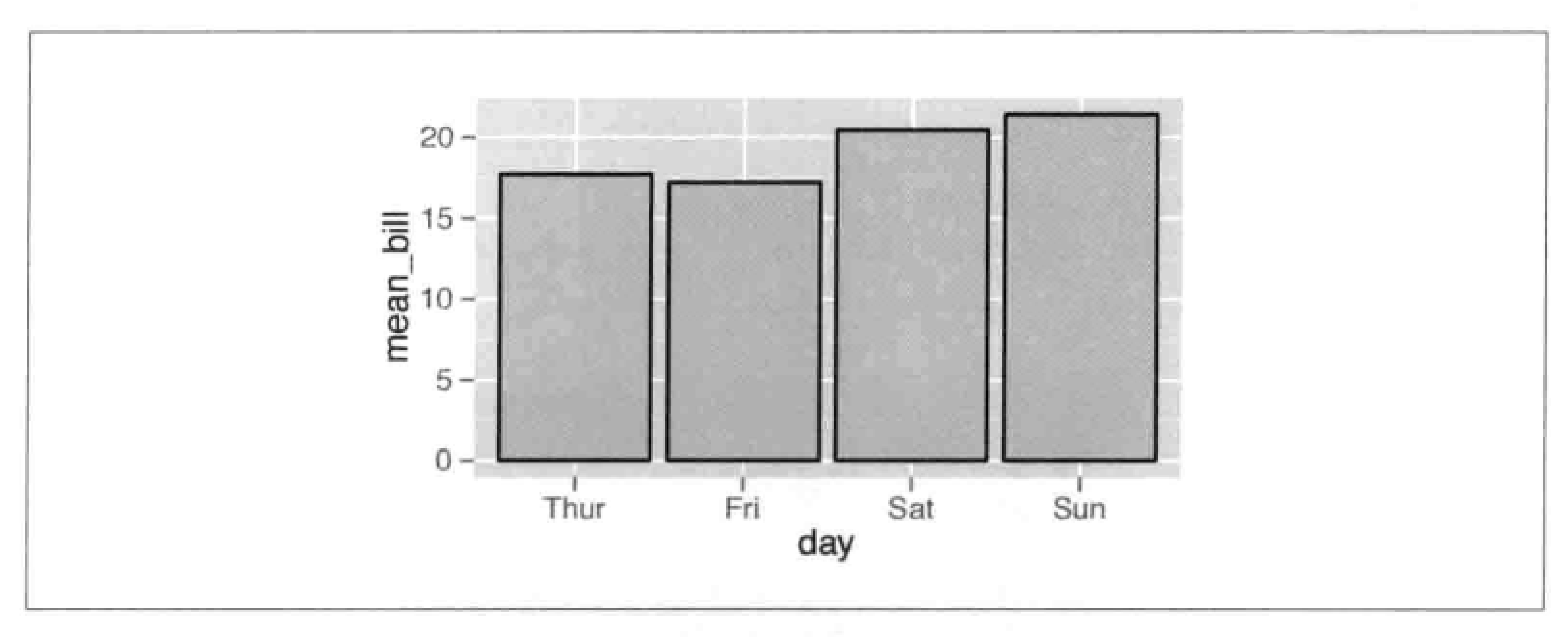


图 3-3　所有条形的填充色和边框线颜色均为单色

```
ggplot(pg_mean, aes(x=group, y=weight)) +
    geom_bar(stat="identity", fill="lightblue", colour="black")
```



在 ggplot2 中，颜色参数默认使用的是英式拼写 colour，而非美式拼写 color。然而，ggplot2 会在底层将美式拼写重映射为英式拼写，因此输入美式拼写的参数并不影响函数的运行。

另见

如果你想让条形图的高度对应于每组变量的频数，可参见 3.3 节的内容。

根据另一个变量值重排因子水平顺序的内容可参见 15.9 节。手动更改因子水平顺序的内容，可参见 15.8 节。

更多关于图形着色的内容，可参见本书第 12 章。

3.2 绘制簇状条形图

问题

如何绘制基于某个分类变量的簇状条形图？

方法

将分类变量映射到 fill 参数，并运行命令 geom_bar(position="dodge")。

下面以 cabbage_exp 数据集为例演示一下绘图过程，cabbage_exp 数据集包含两个分类变量 Cultivar 和 Date 及一个连续型变量 Weight。

```
library(gcookbook) # 为了使用数据
cabbage_exp
```

Cultivar	Date	Weight
c39	d16	3.18
c39	d20	2.80
c39	d21	2.74
c52	d16	2.26
c52	d20	3.11
c52	d21	1.47

我们分别将 Date 和 Cultivar 映射给 x 和 fill（见图 3-4）。

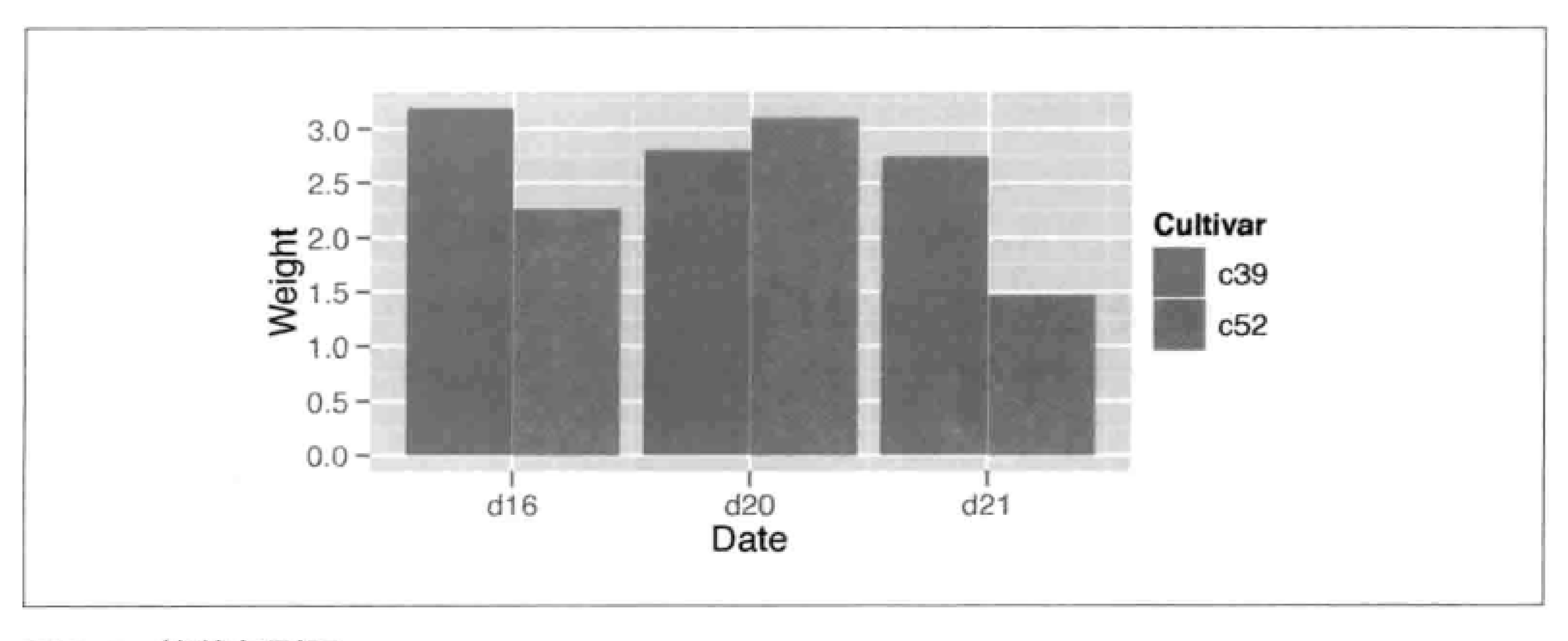


图 3-4　簇状条形图

```
ggplot(cabbage_exp, aes(x=Date, y=Weight, fill=Cultivar)) +
    geom_bar(position="dodge",stat="identity")
```

讨论

最简单的条形图通常只对应一个绘制在 x 轴上的分类变量和一个绘制在 y 轴上的连续型变量。有时候，我们想额外添加一个分类变量跟 x 轴上的分类变量一起对数据进行分组。此时，可通过将该分类变量映射给 fill 参数来绘制簇状条形图，这里的 fill 参数用来指定条形的填充色。在这一过程中必须令参数 position="dodge" 以使得两组

条形在水平方向上错开排列，否则，系统会输出堆积条形图（参见 3.7 节）。

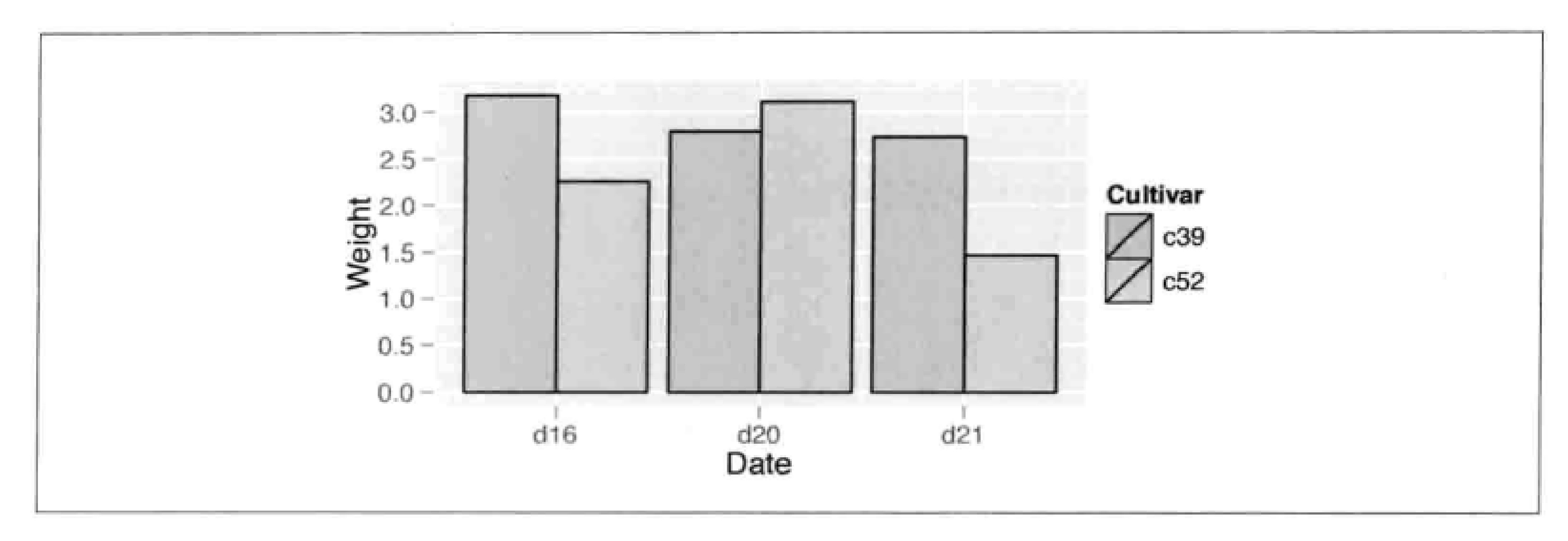


图 3-5 添加了黑色边框线的簇状条形图，这里用了新的调色板

与映射给条形图 *x* 轴的变量类似，映射给条形填充色参数的变量应该是分类变量而不是连续型变量。

我们可以通过将 geom_bar() 中的参数指定为 colour="black" 为条形添加黑色边框线；可以通过 scale_fill_brewer() 或者 scale_fill_manual() 函数对图形颜色进行设置。在图 3-5 中，我们使用 RColorBrewer 包中的 Pastel1 调色盘对图形进行调色。

```
ggplot(cabbage_exp, aes(x=Date, y=Weight, fill=Cultivar)) +
    geom_bar(position="dodge", stat="identity", colour="black") +
    scale_fill_brewer(palette="Pastel1")
```

其他图形属性诸如颜色 colour（指定条形图的边框线颜色）和线型（linestyle）也能用来对变量进行分组，不过，填充色（fill）也许是最合人心意的图形属性。

注意，如果分类变量各水平的组合中有缺失项，那么，绘图结果中的条形则相应地略去不绘，同时，临近的条形将自动扩充到相应位置。删去上例数据中的最后一行后，可得到图 3-6。

```
ce <- cabbage_exp[1:5,]  # 复制删除了最后一行的数据集
ce

Cultivar Date Weight
     c39  d16   3.18
     c39  d20   2.80
     c39  d21   2.74
     c52  d16   2.26
     c52  d20   3.11

ggplot(ce, aes(x=Date, y=Weight, fill=Cultivar)) +
    geom_bar(position="dodge",stat="identity", colour="black") +
    scale_fill_brewer(palette="Pastel1")
```

如果你的数据与上面类似，那么，你可以在分类变量组合缺失的那一项为变量 y 手动输入一个 NA 值。

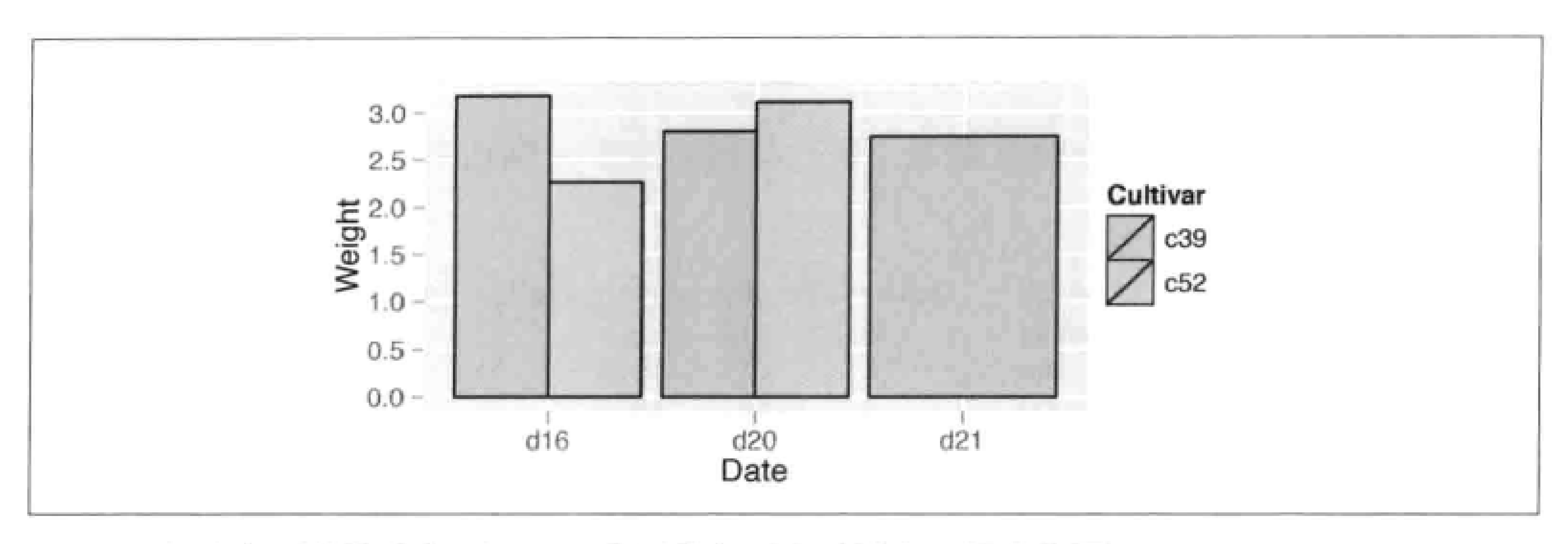


图 3-6　缺失条形的簇状条形图——临近的条形自动扩充到相应位置

另见

更多关于条形图着色的内容，可参见 3.4 节。

根据另一个变量值重排因子水平顺序的内容可参见 15.9 节。

3.3　绘制频数条形图

问题

如果数据集中每行数据对应于一个样本，如何针对样本频数绘制条形图？

方法

使用 geom_bar() 函数，同时不要映射任何变量到 y 参数（见图 3-7）。

```
ggplot(diamonds, aes(x=cut)) + geom_bar()
# 等价于使用 geom_bar(stat="bin")
```

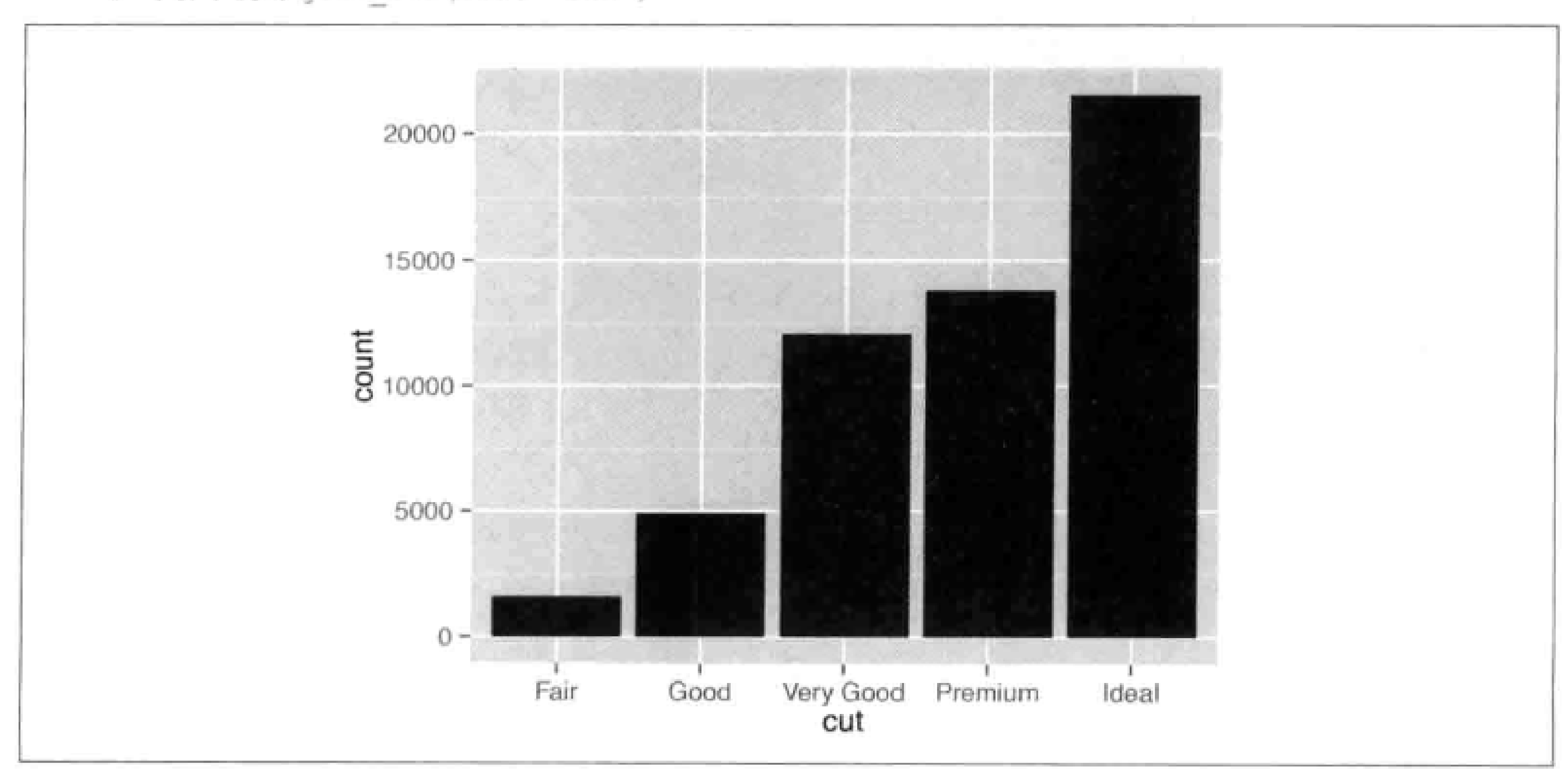


图 3-7　频数条形图

讨论

diamonds 数据集共有 53 940 行数据，每行数据对应于一颗钻石的品质信息：

```
diamonds

        carat       cut  color clarity  depth  table  price     x     y     z
1        0.23     Ideal      E     SI2   61.5     55    326  3.95  3.98  2.43
2        0.21   Premium      E     SI1   59.8     61    326  3.89  3.84  2.31
3        0.23      Good      E     VS1   56.9     65    327  4.05  4.07  2.31
...
53539    0.86   Premium      H     SI2   61.0     58   2757  6.15  6.12  3.74
53540    0.75     Ideal      D     SI2   62.2     55   2757  5.83  5.87  3.64
```

geom_bar() 函数在默认情况下将参数设定为 stat="bin"，该操作会自动计算每组（根据 *x* 轴上面的变量进行分组）变量对应的观测数。从图中可以看到，切工精美的钻石大概有 23 000 颗。

本例中，*x* 轴对应的是离散型变量。当 *x* 轴对应于连续型变量时，我们会得到一张直方图，如图 3-8 所示。



```
ggplot(diamonds, aes(x=carat)) + geom_bar()
```

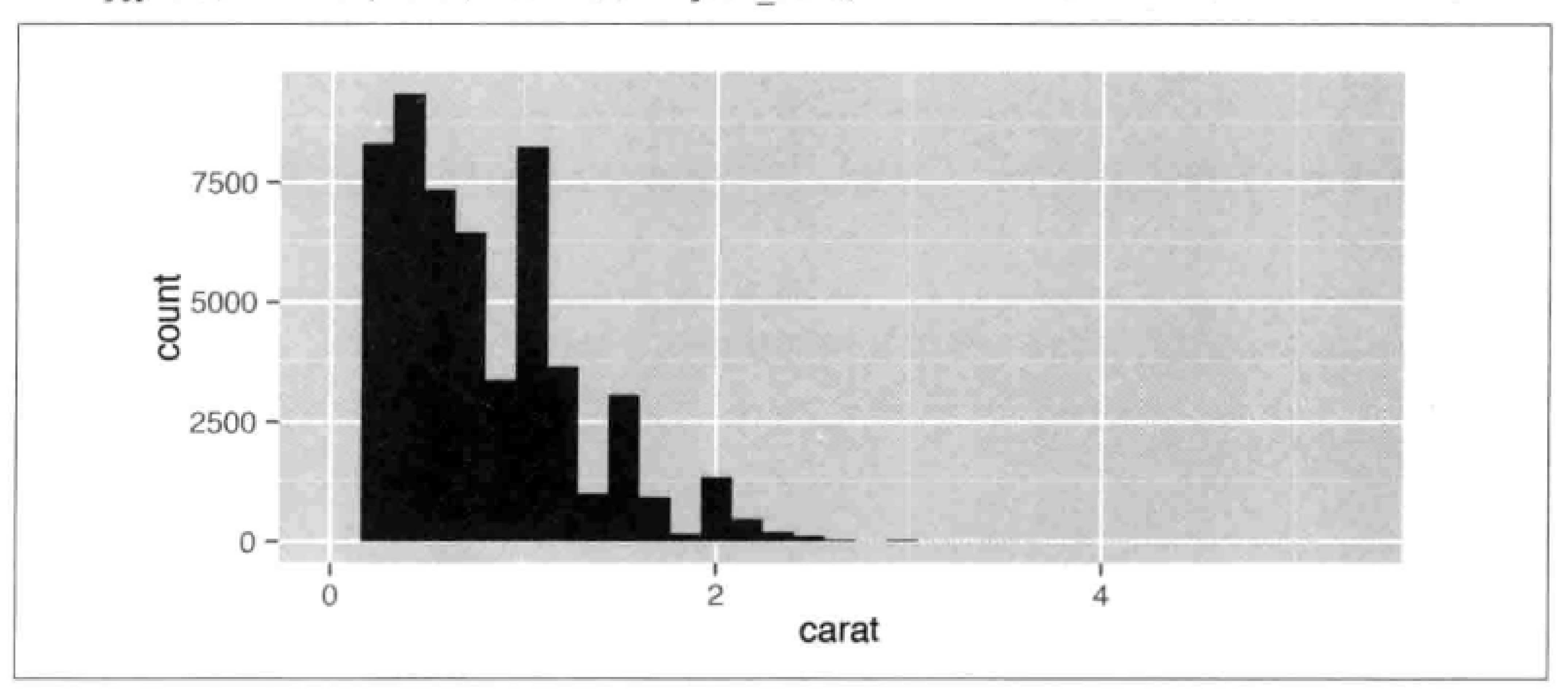


图 3-8　*x* 轴对应于连续型变量的条形图，也即常说的直方图

在这个例子中，使用 geom_bar() 和 geom_histogram() 具有相同的效果。

另见

如果不想让 ggplot() 函数自动计算每组数据的行数绘制频数条形图，而是想通过数据框中的某列来指定 y 参数的话，可以参见 3.1 节的内容。

当然，也可以通过先计算出每组数据的行数，再将计算结果传递给 ggplot() 函数来绘制上图。更多关于数据描述的内容，可参见 15.17 节。

更多关于直方图的内容，可参见 6.1 节。

3.4 条形图着色

问题

如何将条形图中的条形设定为不同的颜色？

方法

将合适的变量映射到填充色（`fill`）上即可。

这里以数据集 `uspopchange` 为例。该数据集描述了美国各州人口自 2000 年到 2010 年的变化情况。我们选取出人口增长最快的十个州进行绘图。图中会根据地区信息（东北部、南部、中北部、西部）对条形进行着色。

首先，选取出人口增长最快的十个州：

```
library(gcookbook)  # 为了使用数据
upc <- subset(uspopchange, rank(Change)>40)
upc
```

```
            State Abb Region Change
          Arizona  AZ   West   24.6
         Colorado  CO   West   16.9
          Florida  FL  South   17.6
          Georgia  GA  South   18.3
            Idaho  ID   West   21.1
           Nevada  NV   West   35.1
   North Carolina  NC  South   18.5
   South Carolina  SC  South   15.3
            Texas  TX  South   20.6
             Utah  UT   West   23.8
```

接下来，将 `Region` 映射到 `fill` 并绘制条形图（见图 3-9）：



```
ggplot(upc, aes(x=Abb, y=Change, fill=Region)) + geom_bar(stat="identity")
```

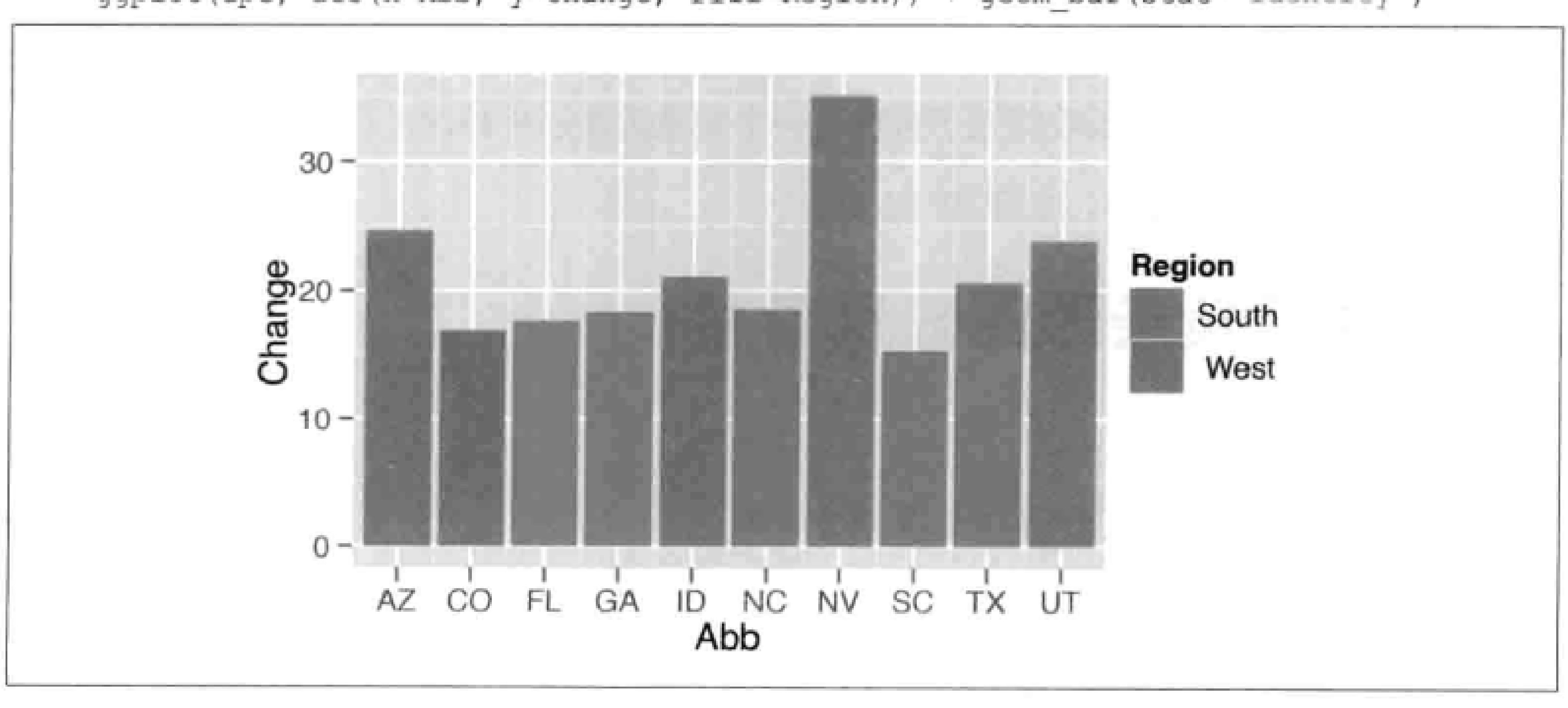


图 3-9　将分类变量映射给 `fill` 参数

讨论

条形图的默认颜色不太吸引眼球，因此，可能需要借助函数 scale_fill_brewer() 或 scale_fill_manual() 重新设定图形颜色。这里我们调用后者。我们通过把参数指定为 colour="black" 将条形的边框线设定为黑色（见图 3-10）。注意：颜色的映射设定是在 aes() 内部完成的，而颜色的重新设定是在 aes() 外部完成的：

```
ggplot(upc, aes(x=reorder(Abb, Change), y=Change, fill=Region)) +
    geom_bar(stat="identity", colour="black") +
    scale_fill_manual(values=c("#669933","#FFCC66")) +
    xlab("State")
```

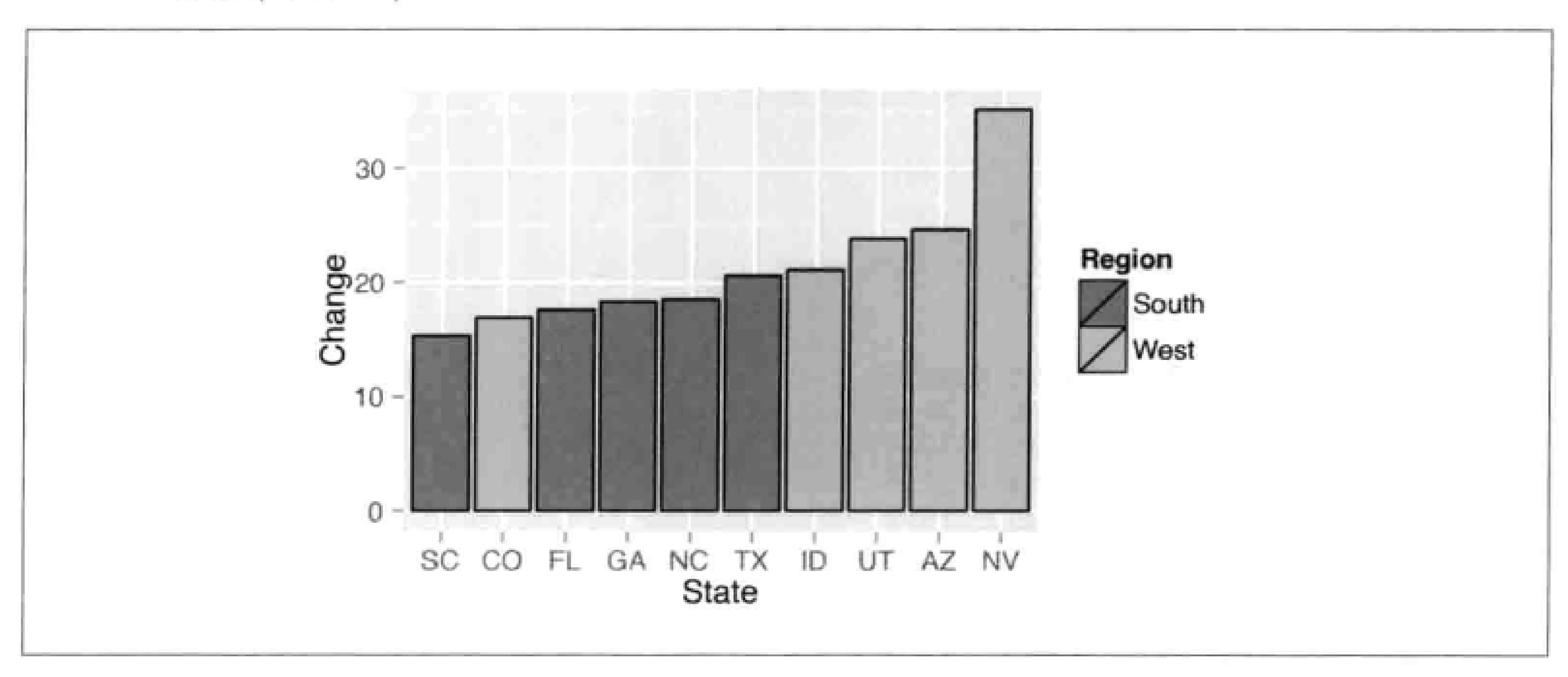


图 3-10　分类着色、具有黑色边框线的簇状条形图，条形根据人口变动百分比排序

本例用到了 reorder() 函数。在本例中，根据条形图的高度进行排序比按照字母顺序对分类变量排序更有意义。

另见

更多关于使用 reorder() 函数将因子根据另一个变量重新水平排序的内容，可参见 15.9 节。

更多关于图形着色的内容，参见第 12 章。

3.5　对正负条形图分别着色

问题

如何根据条形对应的正负取值对其进行分别着色？

方法

下面以 climate 数据的一个子集为例。首先，创建一个对取值正负情况进行标示的变量 pos：

```
library(gcookbook) # 为了使用数据
csub <- subset(climate, Source=="Berkeley" & Year >= 1900)
csub$pos <- csub$Anomaly10y >=0

csub

  Source    Year Anomaly1y Anomaly5y Anomaly10y Unc10y
Berkeley    1900        NA        NA     -0.171  0.108 FALSE
Berkeley    1901        NA        NA     -0.162  0.109 FALSE
Berkeley    1902        NA        NA     -0.177  0.108 FALSE
  ...
Berkeley    2002        NA        NA      0.856  0.028  TRUE
Berkeley    2003        NA        NA      0.869  0.028  TRUE
Berkeley    2004        NA        NA      0.884  0.029  TRUE
```

上述过程准备完毕后，将 pos 映射给填充色参数（fill）并绘制条形图（见图 3-11）。注意：这里条形图的参数设定为 position="identity"，可以避免系统因对负值绘制堆积条形而发出的警告信息。

```
ggplot(csub, aes(x=Year, y=Anomaly10y, fill=pos))+
    geom_bar(stat="identity", position="identity")
```

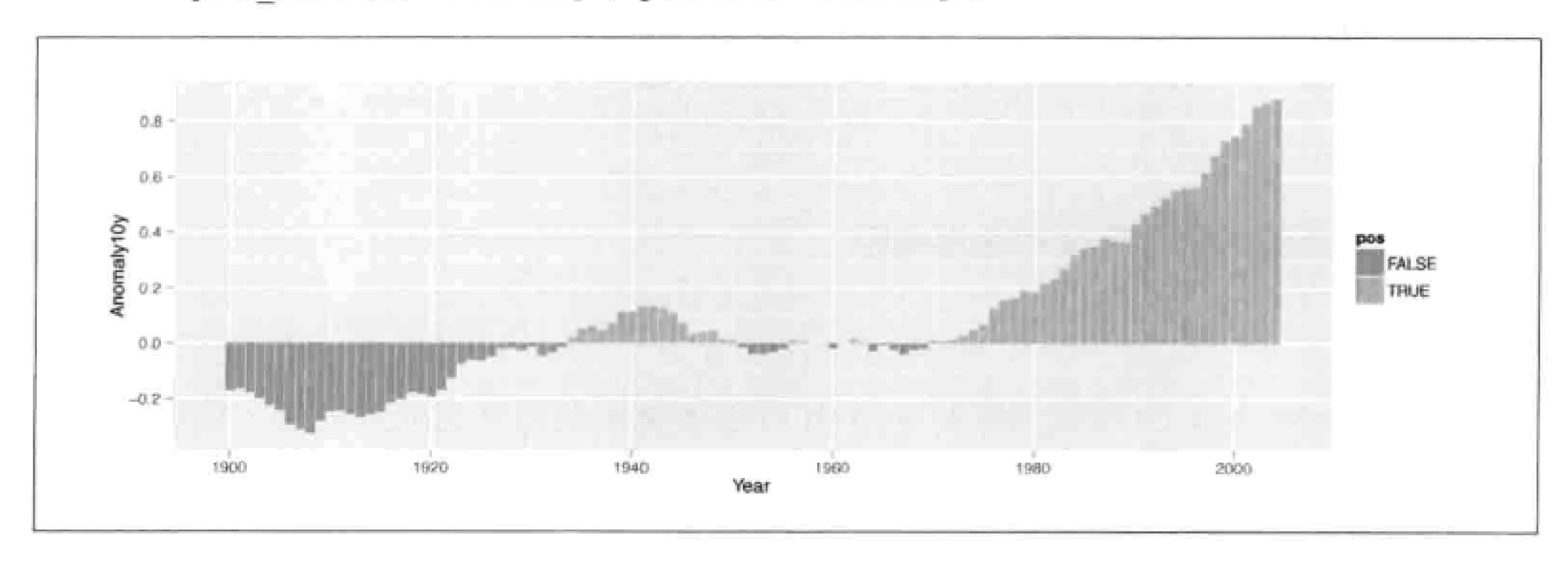


图 3-11 对正负取值的条形分别着色

讨论

上面的绘图过程存在一些问题。首先，图形着色效果可能跟我们想要的相反：蓝色是冷色，通常对应于负值；红色是暖色，通常对应于正值。其次，图例显得多余且扰乱视觉。

我们可以通过 scale_fill_manual() 参数对图形颜色进行调整，设定参数 guide=FALSE 可以删除图例，如图 3-12 所示。同时，我们通过设定边框颜色（colour）和边框线宽度（size）为图形填加一个细黑色边框。其中，边框线宽度（size）是用来控制边框线宽度的参数，单位是毫米：

```
ggplot(csub, aes(x=Year, y=Anomaly10y, fill=pos)) +
    geom_bar(stat="identity", position="identity", colour="black", size=0.25) +
    scale_fill_manual(values=c("#CCEEFF","#FFDDDD"), guide=FALSE)
```

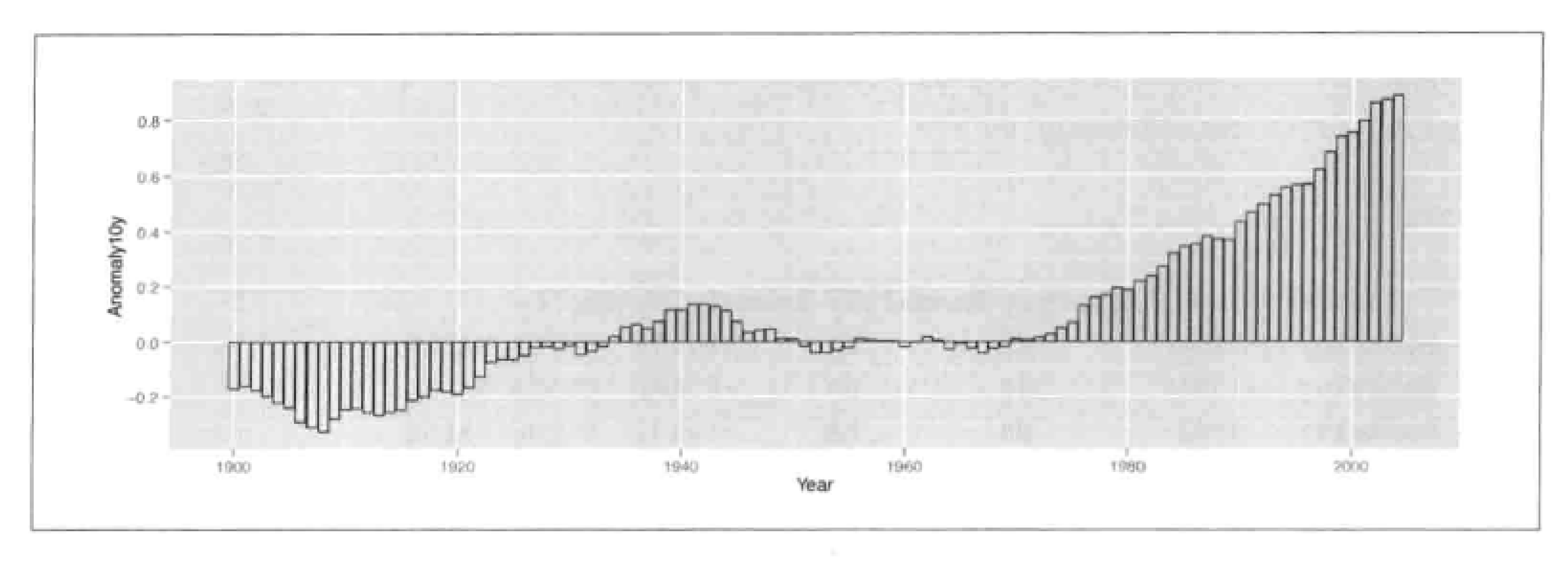


图 3-12 重新设定颜色并移除了图例的条形图

另见

更多关于更改图形颜色的内容可参见 12.3 节和 12.4 节。

更多关于隐藏图例的内容可参见 10.1 节。

3.6 调整条形宽度和条形间距

问题

如何调整条形图的条形宽度和条形间距？

方法

通过设定 geom_bar() 函数的参数 width 可以使条形变得更宽或者更窄。该参数的默认值为 0.9；更大的值将使绘制的条形更宽，反之则是更窄（见图 3-13）。

例如，标准宽度的条形图如下：

```
library(gcookbook) # 为了使用数据

ggplot(pg_mean, aes(x=group, y=weight)) + geom_bar(stat="identity")
```

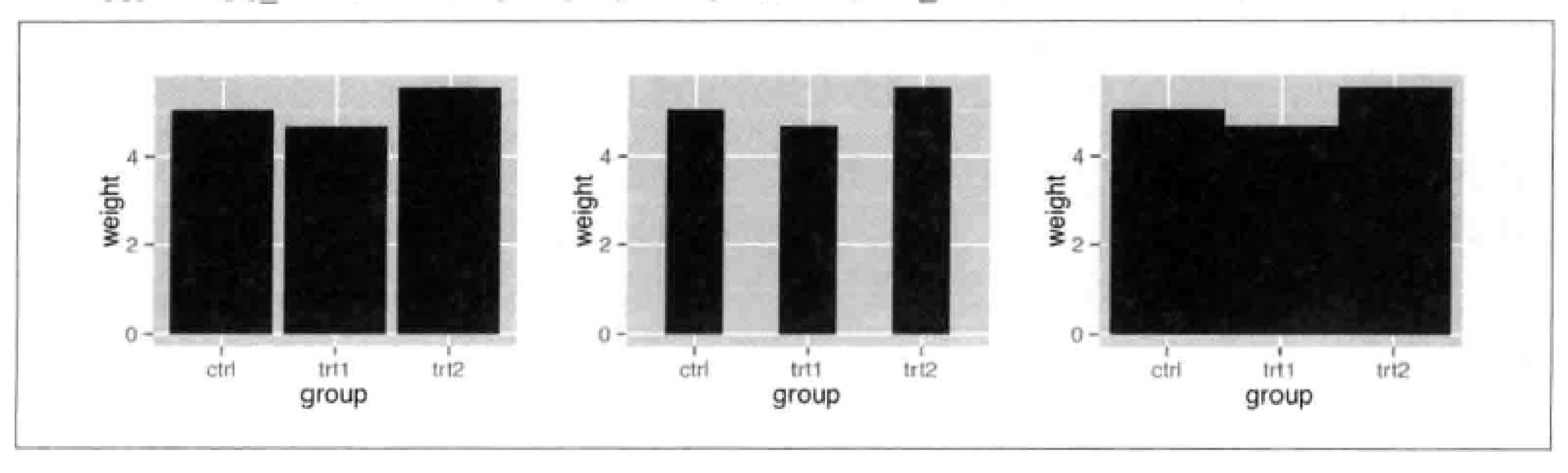


图 3-13 对应于不同条形宽度的条形图

窄些的条形图：

```
ggplot(pg_mean, aes(x=group, y=weight)) + geom_bar(stat="identity", width=0.5)
```

宽些的条形图（条形图的最大宽度为 1）：

```
ggplot(pg_mean, aes(x=group, y=weight)) + geom_bar(stat="identity", width=1)
```

簇状条形图默认组内的条形间距为 0。如果希望增加组内条形的间距，则可以通过将 width 设定得小一些，并令 position_dodge 的取值大于 width（见图 3-14）。

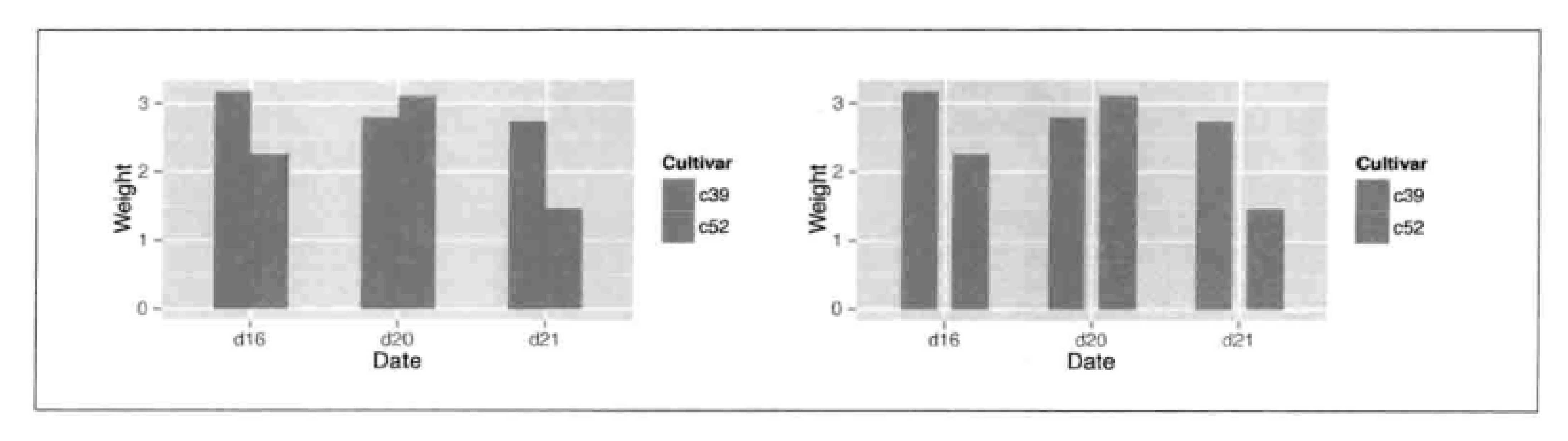


图 3-14　左图：条形更窄的簇状条形图　右图：具有条形间距的簇状条形图

更窄的簇状条形图可运行：

```
ggplot(cabbage_exp, aes(x=Date, y=Weight, fill=Cultivar)) +
    geom_bar(stat="identity", width=0.5, position="dodge")
```

添加条形组距可运行：

```
ggplot(cabbage_exp, aes(x=Date, y=Weight, fill=Cultivar)) +
    geom_bar(stat="identity", width=0.5, position=position_dodge(0.7))
```

第一幅图的绘图命令中用到了参数 position="dodge"，第二幅图的绘图命令中用到的参数是 position=position_dodge()。这是因为 position="dodge" 是参数默认为 0.9 的 position_dodge() 的简写。当我们需要单独指定该参数的时候，必须输入完整的命令。

讨论

width 参数的默认值是 0.9，position_dodge 函数中 width 参数的默认值也是 0.9。更确切地说，position_dodge 函数和 geom_bar() 函数中的 width 参数的取值是一样的。

下面的四个命令是等价的：

```
geom_bar(position="dodge")
geom_bar(width=0.9, position=position_dodge())
geom_bar(position=position_dodge(0.9))
geom_bar(width=0.9, position=position_dodge(width=0.9))
```

条形图中，条形中心对应的 x 轴坐标分别是 1、2、3 等，但通常我们不会利用上这些数值。当用户运行命令 geom_bar(width=0.9) 时，每组条形将在 x 轴上占据 0.9 个单位宽度。运行命令 position_dodge(width=0.9) 时，ggplot2 会自动调整条形位置，以使每个条形的中心恰好位于当每组条形宽度为 0.9，且组内条形紧贴在一起时的位

置，如图 3-15 所示。图中上下两部分都对应 position_dodge(width=0.9)，只是上图对应于 0.9 的条形宽度，下图对应于 0.2 的条形宽度。虽然上下两部分对应的条形宽度不同，但两图的条形中心是上下对齐的。

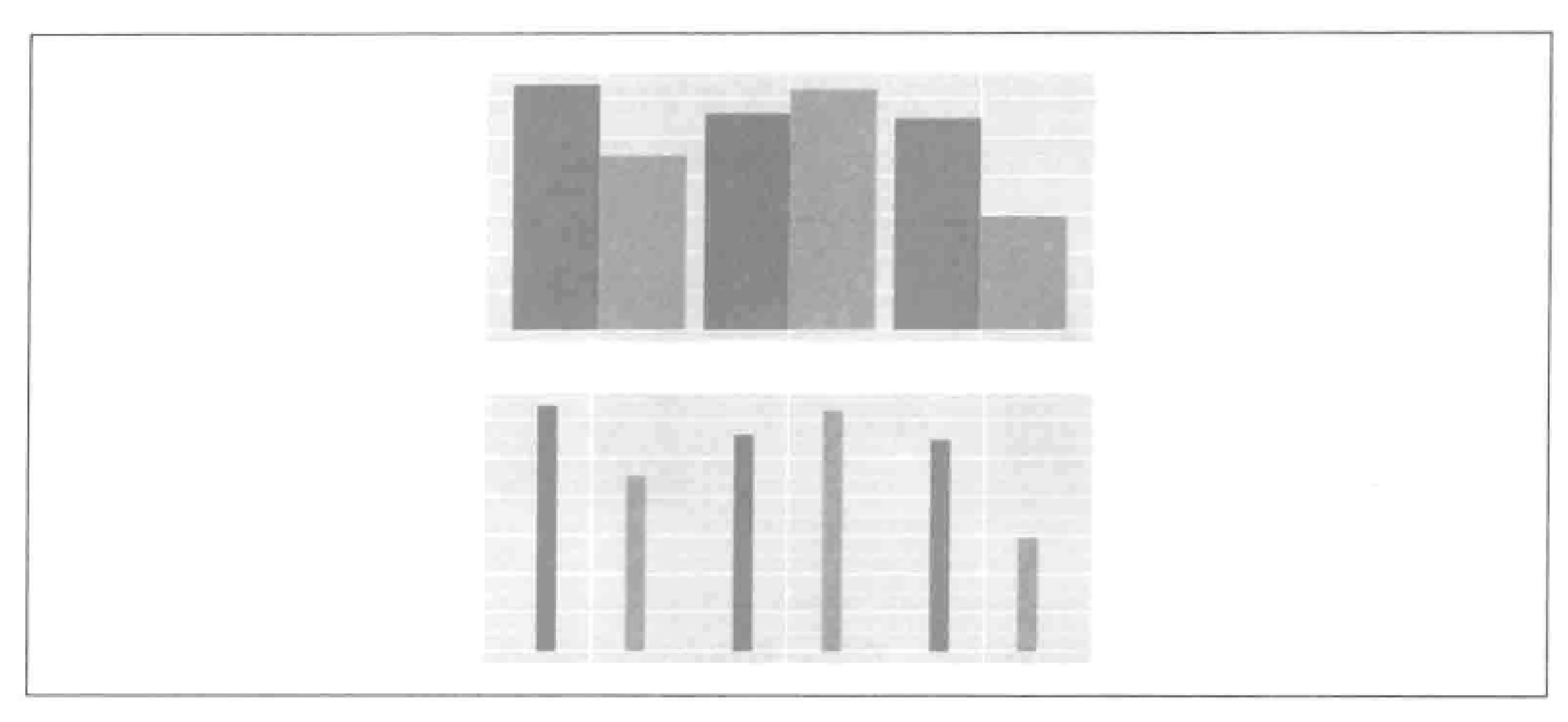


图 3-15　条形间距相同但条形宽度不同的簇状条形图

如果你将整幅图形进行伸缩，条形图也会依照相应的比例进行伸缩。要了解图形是怎样变化的，只需改变图形所在窗口的大小，然后，观察图形的变化即可。更多关于在输出图形文件时控制图片大小的内容可参见第 14 章。

3.7　绘制堆积条形图

问题

如何绘制堆积条形图？

方法

使用 geom_bar() 函数，并映射一个变量给填充色参数（fill）即可。该命令会将 Date 对应到 *x* 轴上，并以 Cultivar 作为填充色，如图 3-16 所示。

```
library(gcookbook) # 为了使用数据
ggplot(cabbage_exp, aes(x=Date, y=Weight, fill=Cultivar)) +
    geom_bar(stat="identity")
```

讨论

弄清楚图形对应的数据结构有助于理解图形的绘制过程。上例数据集中 Date 变量对应于三个水平、Cultivar 变量对应于两个水平，两个变量不同水平的组合又分别与一个 Weight 变量相对应：

```
cabbage_exp

Cultivar  Date Weight        sd  n         se
     c39   d16   3.18 0.9566144 10 0.30250803
     c39   d20   2.80 0.2788867 10 0.08819171
     c39   d21   2.74 0.9834181 10 0.31098410
     c52   d16   2.26 0.4452215 10 0.14079141
     c52   d20   3.11 0.7908505 10 0.25008887
     c52   d21   1.47 0.2110819 10 0.06674995
```

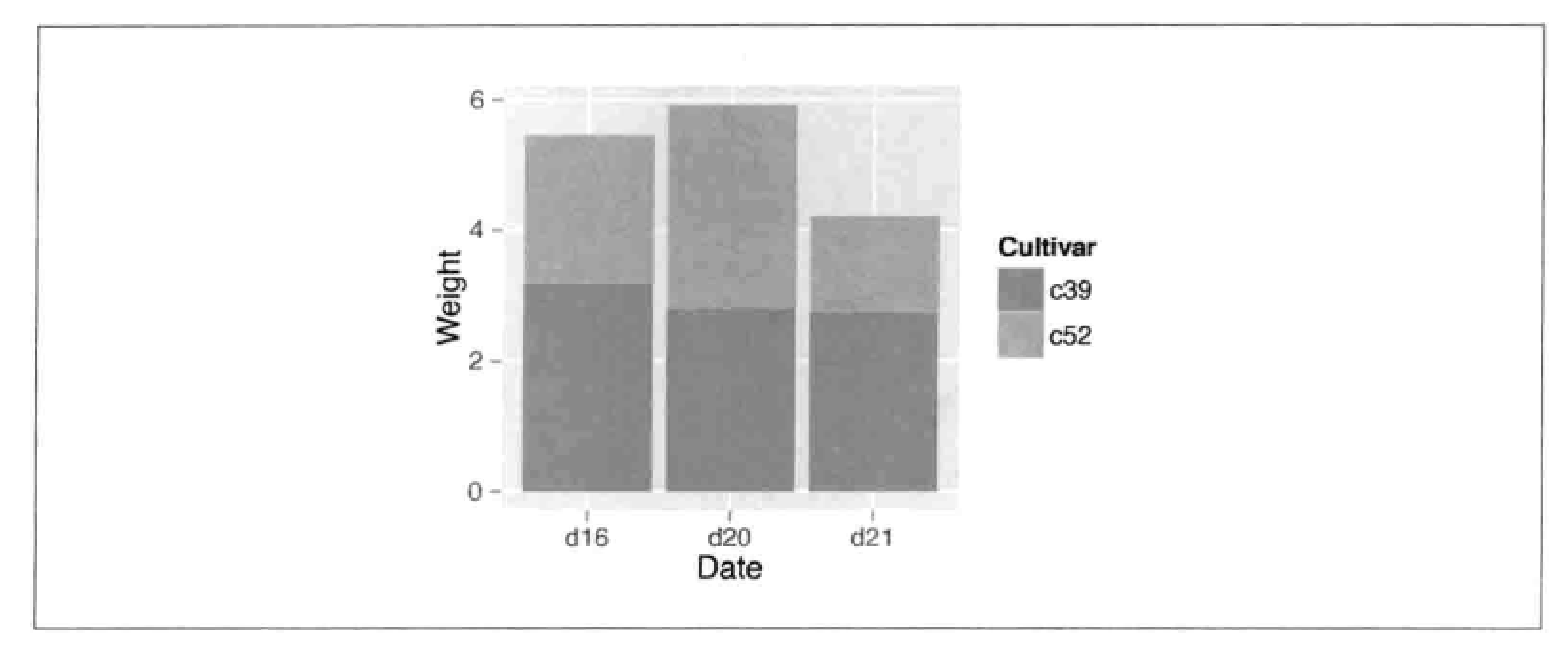


图 3-16　堆积条形图

默认绘制的条形图有一个问题，即条形的堆积顺序与图例顺序是相反的。我们可以通过 guides() 函数对图例顺序进行调整，并指定图例所对应的需要调整的图形属性，本例中对应的是填充色（fill），如图 3-17 所示。

```
ggplot(cabbage_exp, aes(x=Date, y=Weight, fill=Cultivar)) +
    geom_bar(stat="identity") +
    guides(fill=guide_legend(reverse=TRUE))
```

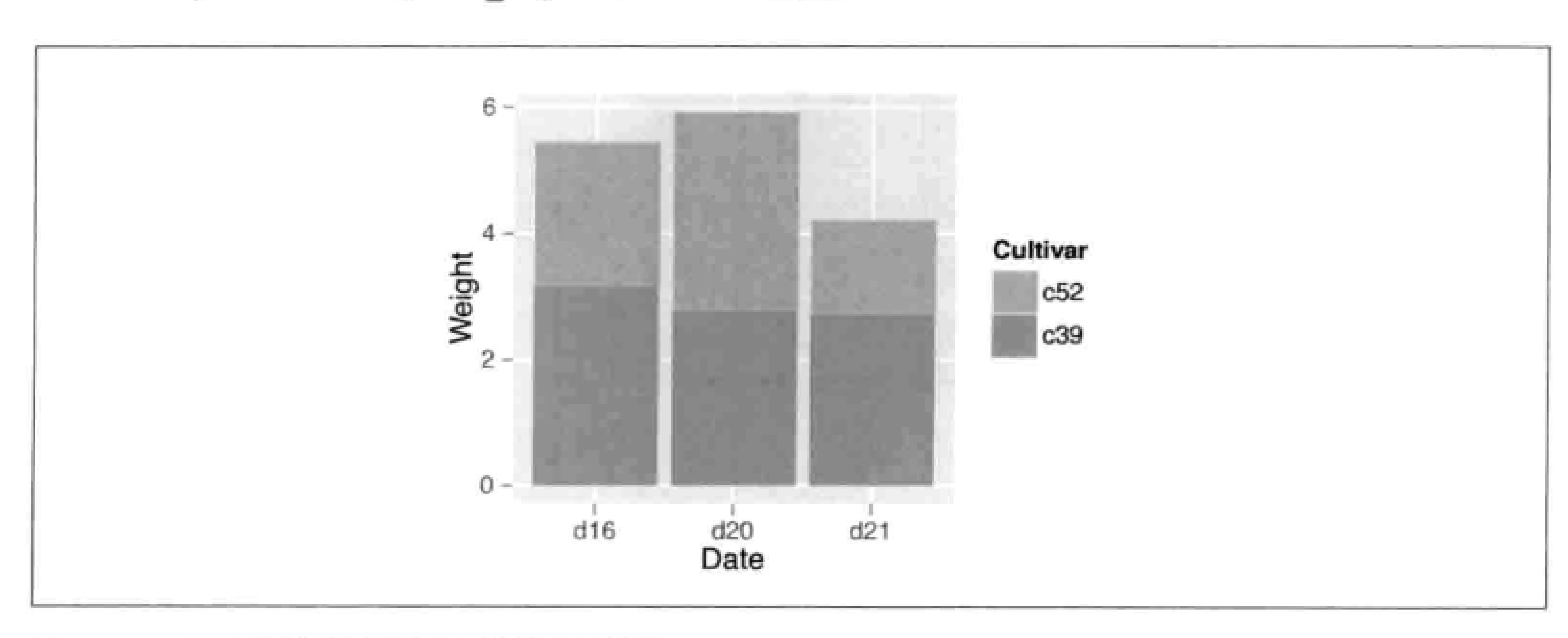


图 3-17　调整图例顺序后的堆积条形图

如果你想调整条形的堆叠顺序，可以通过指定图形映射中的参数 order=desc() 来实现：

```
library(plyr)   # 为了使用 desc() 函数
ggplot(cabbage_exp, aes(x=Date, y=Weight, fill=Cultivar, order=desc(Cultivar))) +
    geom_bar(stat="identity")
```

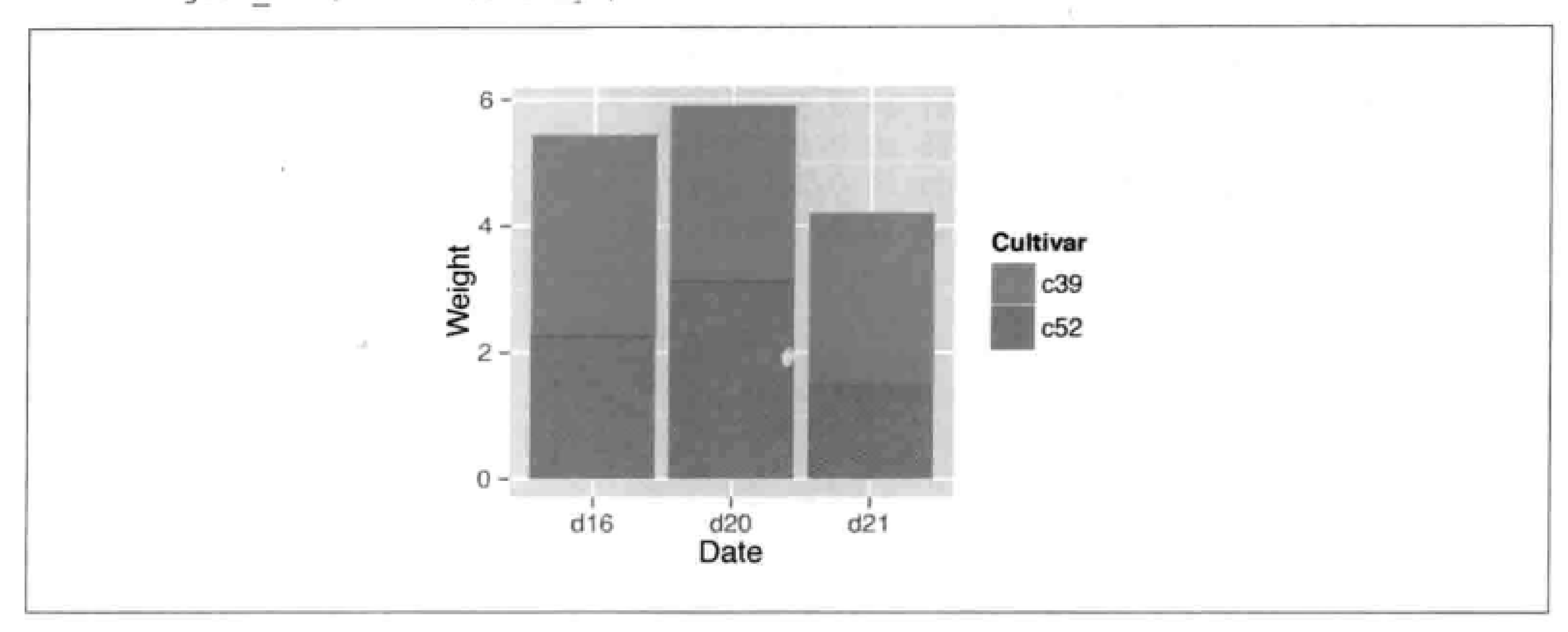


图 3-18　翻转了堆叠顺序的堆积条形图

当然，也可以通过调整数据框中对应列的因子顺序来实现上述操作（参见 15.8 节），但需谨慎进行该操作，因为对数据进行修改可能导致其他分析结果也发生改变。

为了获得效果更好的条形图，我们保持逆序的图例顺序不变，同时，使用 scale_fill_brewer() 函数得到一个新的调色板，最后设定 colour="black" 为条形添加一个黑色边框线（如图 3-19 所示）。

```
ggplot(cabbage_exp, aes(x=Date, y=Weight, fill=Cultivar)) +
    geom_bar(stat="identity", colour="black") +
     guides(fill=guide_legend(reverse=TRUE)) +
     scale_fill_brewer(palette="Pastel1")
```

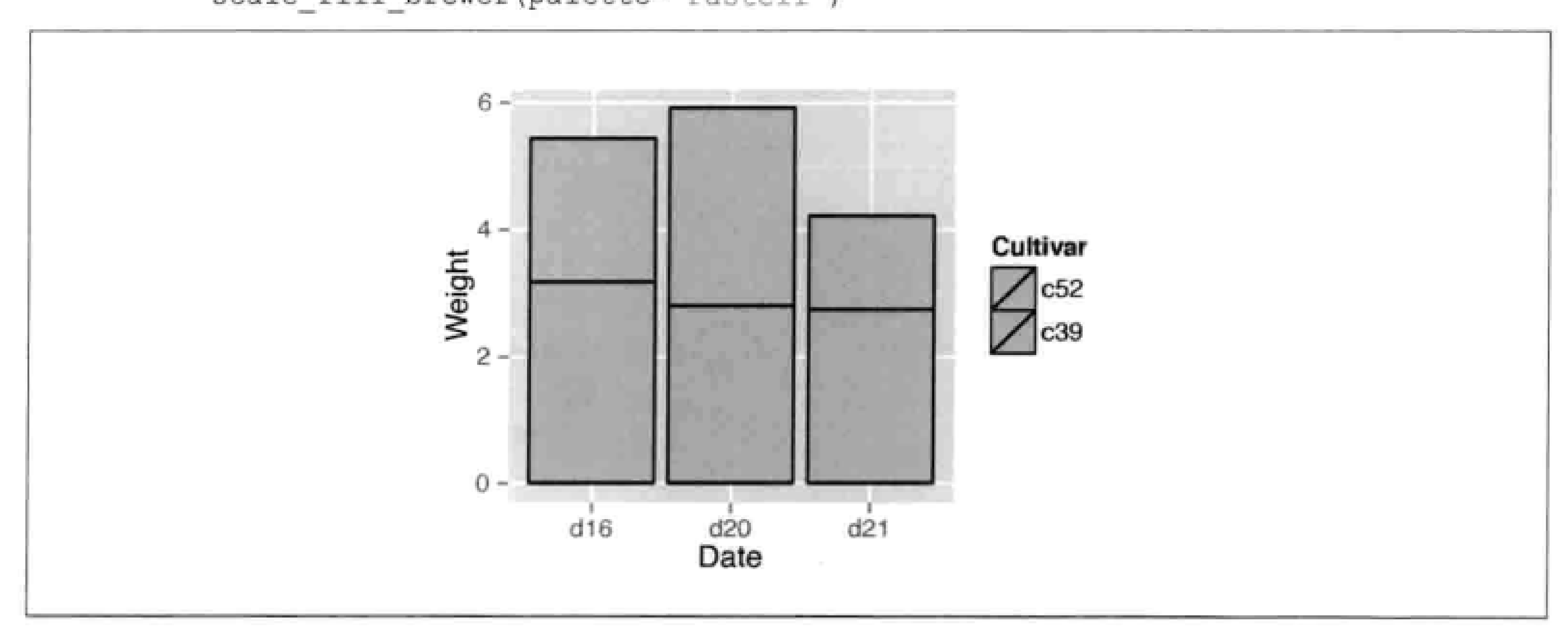


图 3-19　翻转图例顺序，使用新调色板和黑色边框线的堆积条形图

另见

更多关于条形图着色的内容可参见 3.4 节。

将因子根据另一个变量重新排列水平顺序的内容可参见 15.9 节。手动更改因子水平顺序的内容，可参见 15.8 节。

3.8 绘制百分比堆积条形图

问题

如何绘制可展示百分比的堆积条形图（又叫百分比堆积条形图）？

方法

首先，通过 plyr 包中的 ddply() 函数和 transform() 函数将每组条形对应的数据标准化为 100% 格式，之后，针对计算得到的结果绘制堆积条形图即可，如图 3-20 所示。

```
library(gcookbook)  # 为了使用数据
library(plyr)
# 以 Date 为切割变量 () 对每组数据进行 transform()
ce <- ddply(cabbage_exp, "Date", transform,
            percent_weight = Weight / sum(Weight) * 100)

ggplot(ce, aes(x=Date, y=percent_weight, fill=Cultivar)) +
    geom_bar(stat="identity")
```

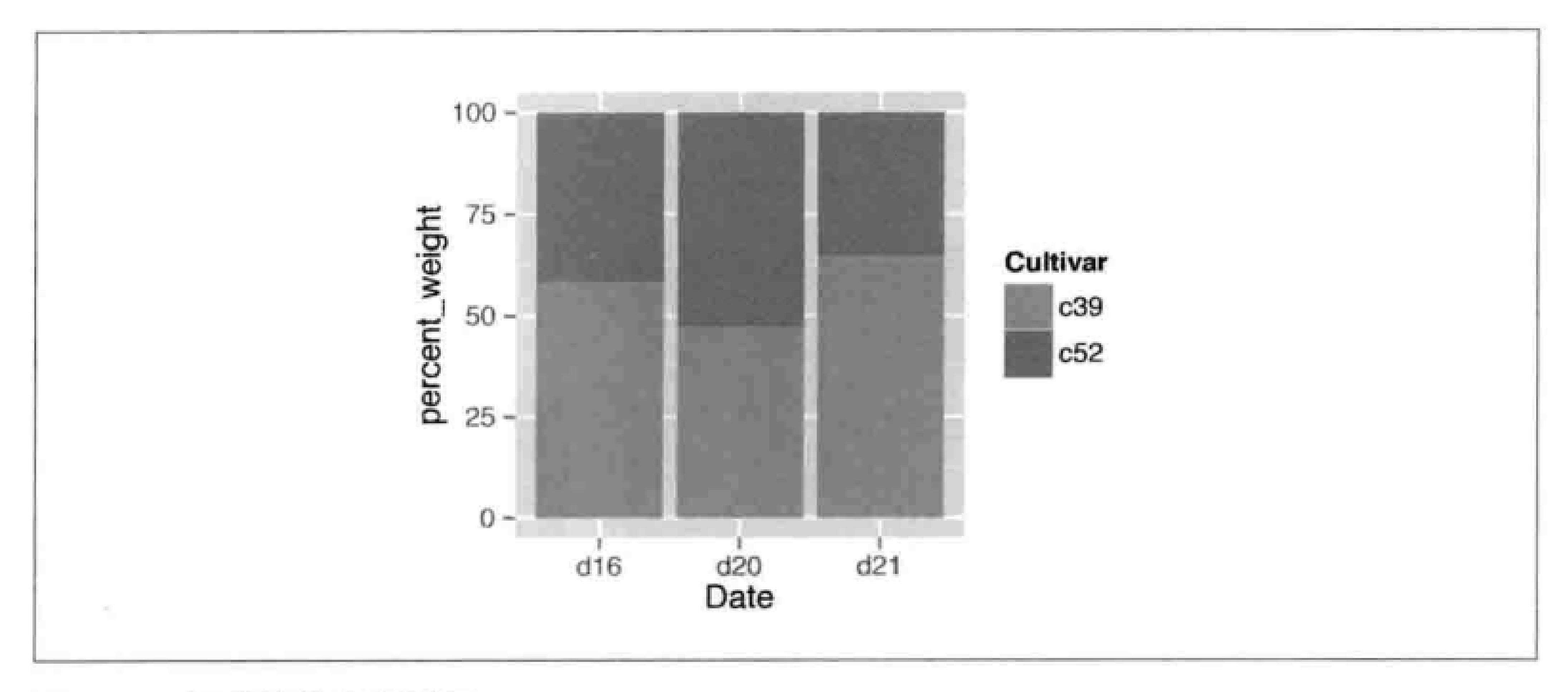


图 3-20　百分比堆积条形图

讨论

我们用 ddply() 函数计算每组 Date 变量对应的百分比。本例中，ddply() 函数根据指定的变量 Date 对数据框 cabbage_exp 进行分组，并对各组数据执行 transform() 函数（ddply() 函数中设定的其他参数也会传递给该函数）。

下面是 cabbage_exp 数据，从中可以看出 ddply() 命令对其进行操作的过程。

```
cabbage_exp

 Cultivar Date Weight        sd  n         se
      c39  d16   3.18 0.9566144 10 0.30250803
      c39  d20   2.80 0.2788867 10 0.08819171
      c39  d21   2.74 0.9834181 10 0.31098410
      c52  d16   2.26 0.4452215 10 0.14079141
      c52  d20   3.11 0.7908505 10 0.25008887
      c52  d21   1.47 0.2110819 10 0.06674995

ce <- ddply(cabbage_exp, "Date", transform,
            percent_weight = Weight / sum(Weight) * 100)

Cultivar Date  Weight        sd  n         se percent_weight
     c39  d16    3.18 0.9566144 10 0.30250803       58.45588
     c52  d16    2.26 0.4452215 10 0.14079141       41.54412
     c39  d20    2.80 0.2788867 10 0.08819171       47.37733
     c52  d20    3.11 0.7908505 10 0.25008887       52.62267
     c39  d21    2.74 0.9834181 10 0.31098410       65.08314
     c52  d21    1.47 0.2110819 10 0.06674995       34.91686
```

计算出百分比之后，就可以按照绘制常规堆积条形图的方法来绘制百分比堆积条形图了。

跟常规堆积条形图一样，我们可以调整百分比堆积条形图的图例顺序、更换调色板及添加边框线，如图 3-21 所示。

```
ggplot(ce, aes(x=Date, y=percent_weight, fill=Cultivar)) +
    geom_bar(stat="identity", colour="black") +
    guides(fill=guide_legend(reverse=TRUE)) +
    scale_fill_brewer(palette="Pastel1")
```

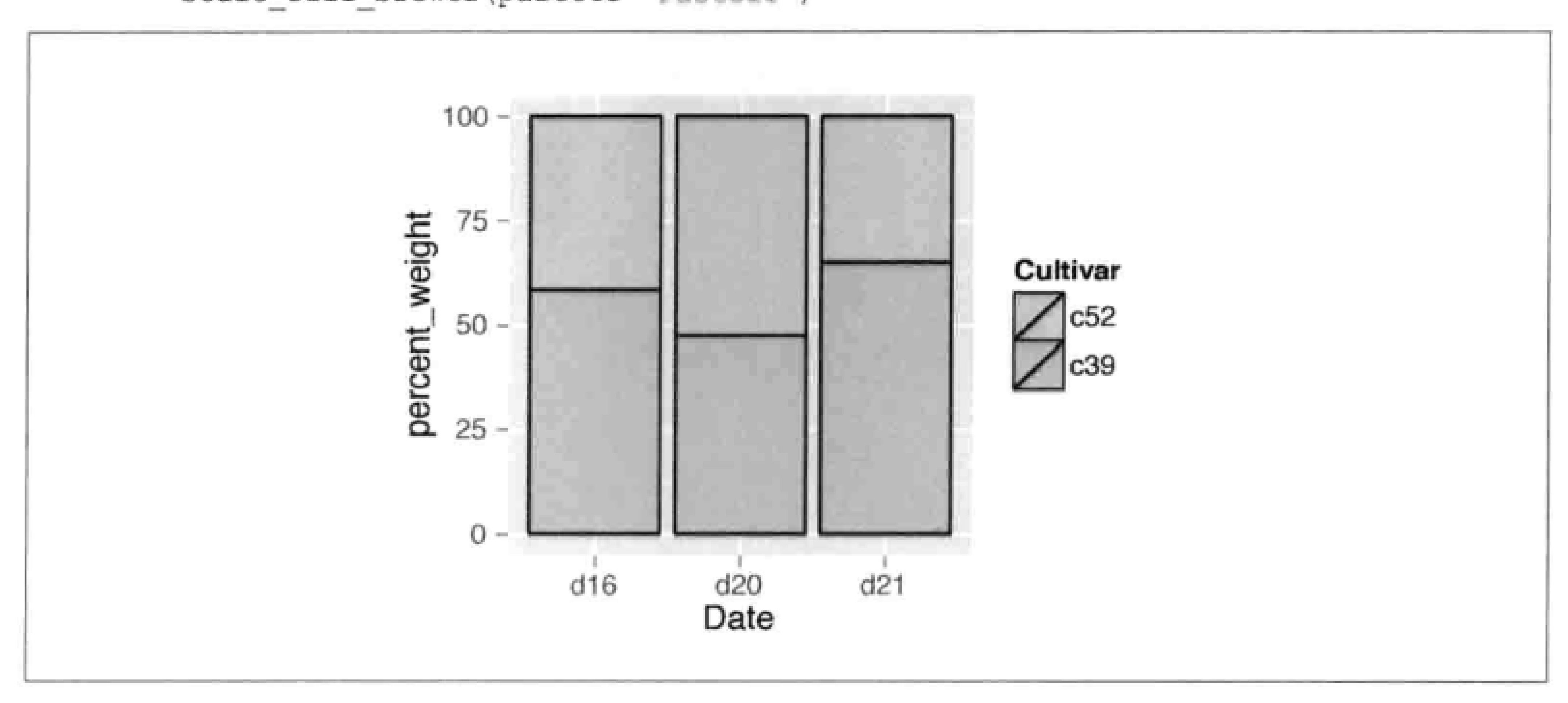


图 3-21　反转图例顺序，使用新调色板和黑色边线框的百分比堆积条形图

参见

更多关于分组对数据进行变换的内容可参见 15.16 节。

3.9 添加数据标签

问题

如何给条形图添加数据标签？

方法

在绘图命令中加上 geom_text() 即可为条形图添加数据标签。运行命令时，需要分别指定一个变量映射给 x、y 和标签本身。通过设定 vjust（竖直调整数据标签位置）可以将标签位置移动至条形图顶端的上方或者下方，如图 3-22 所示。

```
library(gcookbook) # 为了使用数据

# 在条形图顶端下方
ggplot(cabbage_exp, aes(x=interaction(Date, Cultivar), y=Weight)) +
      geom_bar(stat="identity") +
      geom_text(aes(label=Weight), vjust=1.5, colour="white")

# 在条形图顶端上方
ggplot(cabbage_exp, aes(x=interaction(Date, Cultivar), y=Weight)) +
    geom_bar(stat="identity") +
    geom_text(aes(label=Weight), vjust=-0.2)
```

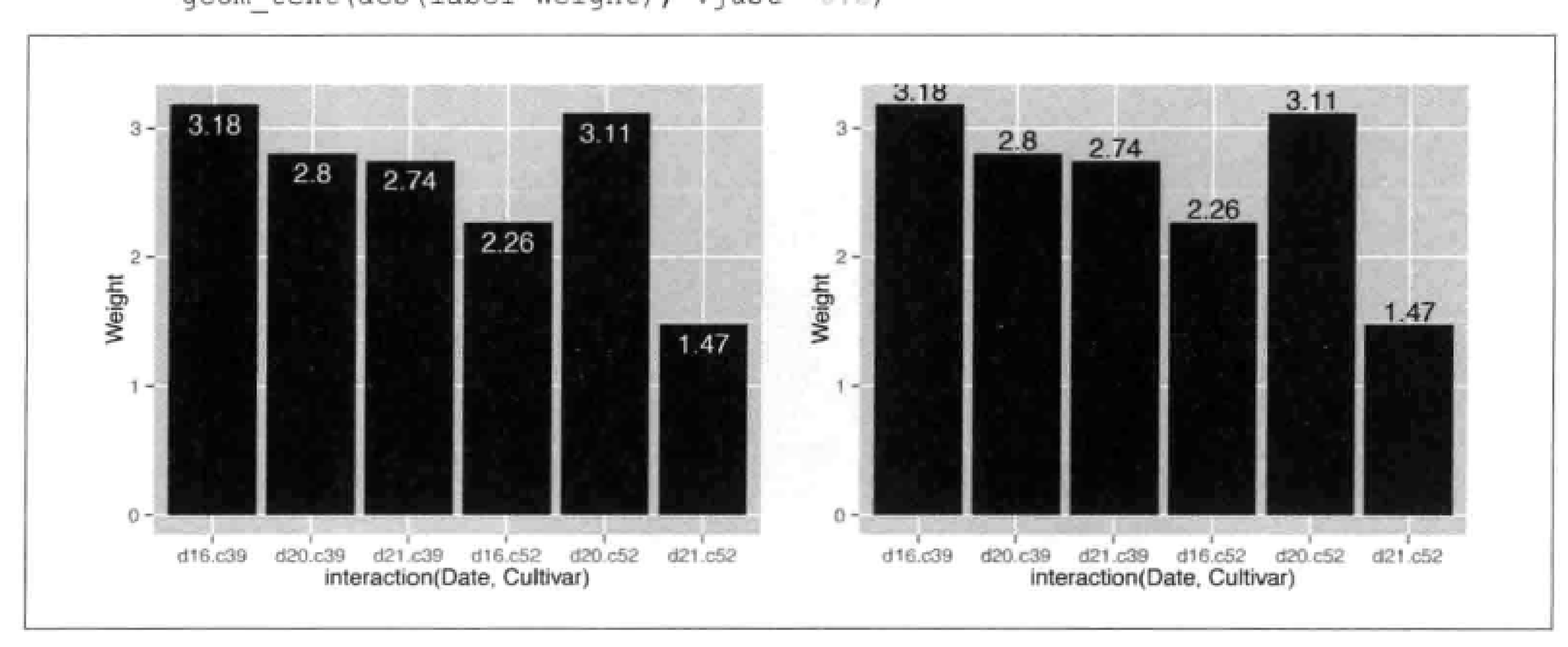


图 3-22 左图：置于条形图顶端下方的数据标签 右图：置于条形图顶端上方的数据标签

注意，当数据标签被置于条形图顶端时，它们可能会被遮挡。为了避免这个问题，可以参见 8.2 节的内容。

讨论

在图 3-22 中，数据标签的 y 轴坐标位于每个条形的顶端中心位置；通过设定竖直调整（vjust）可以将数据标签置于条形图顶端的上方或者下方。这种做法的不足之处在于当数据标签被置于条形图顶端上方时有可能使数据标签溢出绘图区域。为了修正这个

问题，我们可以手动设定 y 轴的范围，也可以保持竖直调整不变，而令数据标签的 y 轴坐标高于条形图顶端。后一种办法的不足之处在于，当你想将数据标签完全置于条形图顶端上方或者下方的时候，竖直方向调整的幅度依赖于 y 轴的数据范围；而更改 vjust 时，数据标签离条形顶端的距离会根据条形图的高度自动进行调整。

```
# 将 y 轴上限变大
ggplot(cabbage_exp, aes(x=interaction(Date, Cultivar), y=Weight)) +
    geom_bar(stat="identity") +
    geom_text(aes(label="Weight"), vjust=-0.2) +
    ylim(0, max(cabbage_exp$Weight)*1.05)

# 设定标签的 y 轴位置使其略高于条形图顶端—y 轴范围会自动调整
ggplot(cabbage_exp, aes(x=interaction(Date, Cultivar), y=Weight)) +
    geom_bar(stat="identity") +
    geom_text(aes(y=Weight+0.1, label=Weight))
```

对于簇状条形图，需要设定 position=position_dodge() 并给其一个参数来设定分类间距。分类间距的默认值是 0.9，因为簇状条形图的条形更窄，所以，需要使用字号（size）来缩小数据标签的字体大小以匹配条形宽度。数据标签的默认字号是 5，这里我们将字号设定为 3 使其看起来更小（见图 3-23）。

```
ggplot(cabbage_exp, aes(x=Date, y=Weight, fill=Cultivar)) +
    geom_bar(stat="identity", position="dodge") +
    geom_text(aes(label=Weight), vjust=1.5, colour="white",
              position=position_dodge(.9), size=3)
```

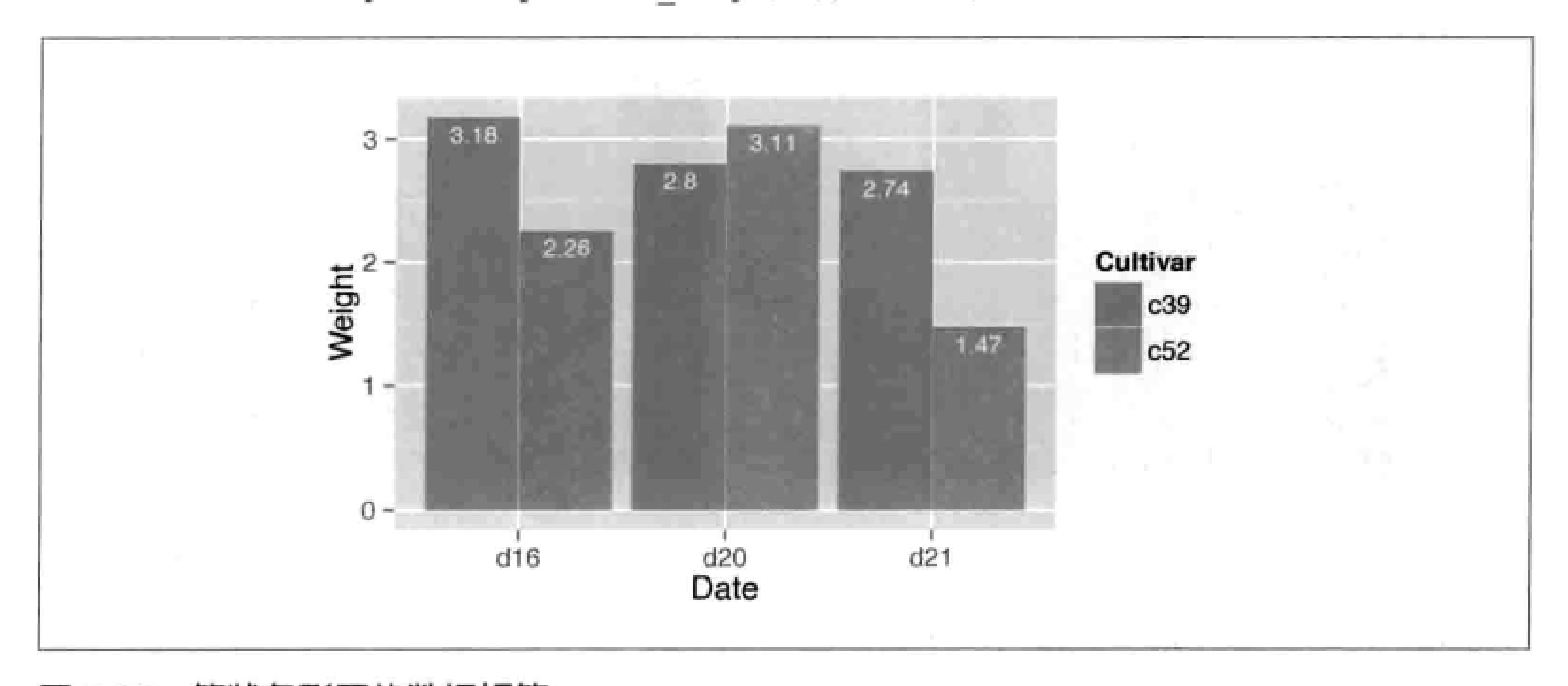


图 3-23　簇状条形图的数据标签

向堆积条形图添加数据标签之前，要先对每组条形对应的数据进行累积求和。在进行本操作之前，须保证数据的合理排序，否则，可能计算出错误的累积和。我们可以用 plyr 包中的 arrange() 函数完成上述操作，plyr 包是一个随 ggplot2 包加载的软件包。

```
library(plyr)
# 根据日期和性别对数据进行排序
ce <- arrange(cabbage_exp, Date, Cultivar)
```

确认数据合理排序之后，我们可以借助 ddply() 函数以 Date 为分组变量对数据进行分组，并分别计算每组数据对应的变量 Weight 的累积和。

```
# 计算累积和
ce <- ddply(ce, "Date", transform, label_y=cumsum(Weight))
ce

Cultivar Date Weight        sd  n         se percent_weight label_y
     c39  d16   3.18 0.9566144 10 0.30250803       58.45588    3.18
     c52  d16   2.26 0.4452215 10 0.14079141       41.54412    5.44
     c39  d20   2.80 0.2788867 10 0.08819171       47.37733    2.80
     c52  d20   3.11 0.7908505 10 0.25008887       52.62267    5.91
     c39  d21   2.74 0.9834181 10 0.31098410       65.08314    2.74
     c52  d21   1.47 0.2110819 10 0.06674995       34.91686    4.21

ggplot(ce, aes(x=Date, y=Weight, fill=Cultivar)) +
    geom_bar(stat="identity") +
    geom_text(aes(y=label_y, label=Weight), vjust=1.5, colour="white")
```

结果如图 3-24 所示。

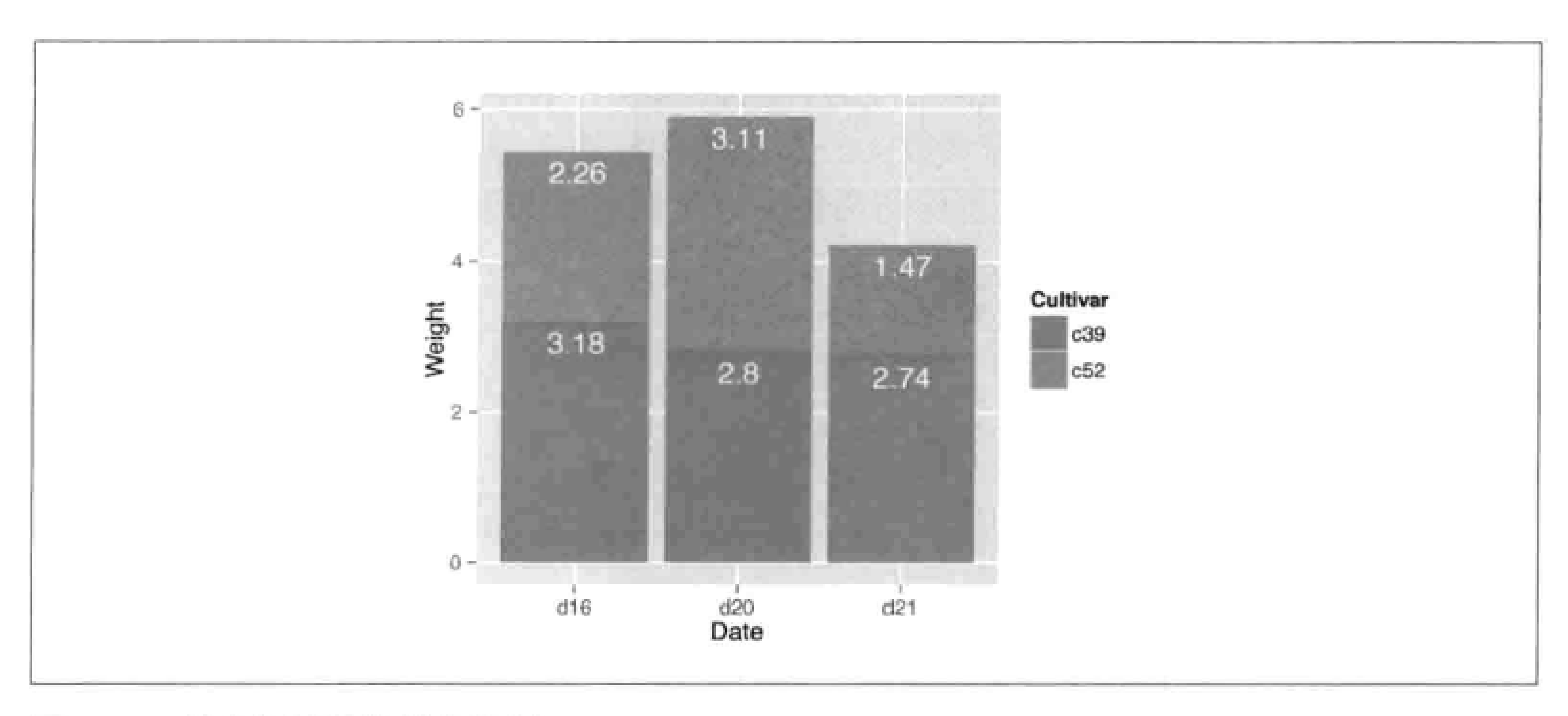


图 3-24　堆积条形图的数据标签

使用数据标签时，对堆叠顺序的调整最好是通过在计算累积和之前修改因子的水平顺序（参见 15.8 节）来完成。另一种修改堆叠顺序的方法是在标度中指定 breaks 参数，但在这里此方法并不合适，因为累计求和的顺序与堆叠的顺序并不一致。

如果想把数据标签置于条形中部（见图 3-25），须对累计求和的结果加以调整，并同时略去 geom_bar() 函数中对 y 偏移量（offset）的设置：

```
ce <- arrange(cabbage_exp, Date, Cultivar)

# 计算 y 轴的位置，将数据标签置于条形中部
ce <- ddply(ce, "Date", transform, label_y=cumsum(Weight)-0.5*Weight)

ggplot(ce, aes(x=Date, y=Weight, fill=Cultivar))+
```

```
    geom_bar(stat="identity")+
    geom_text(aes(y=label_y, label=Weight), colour="White")
```

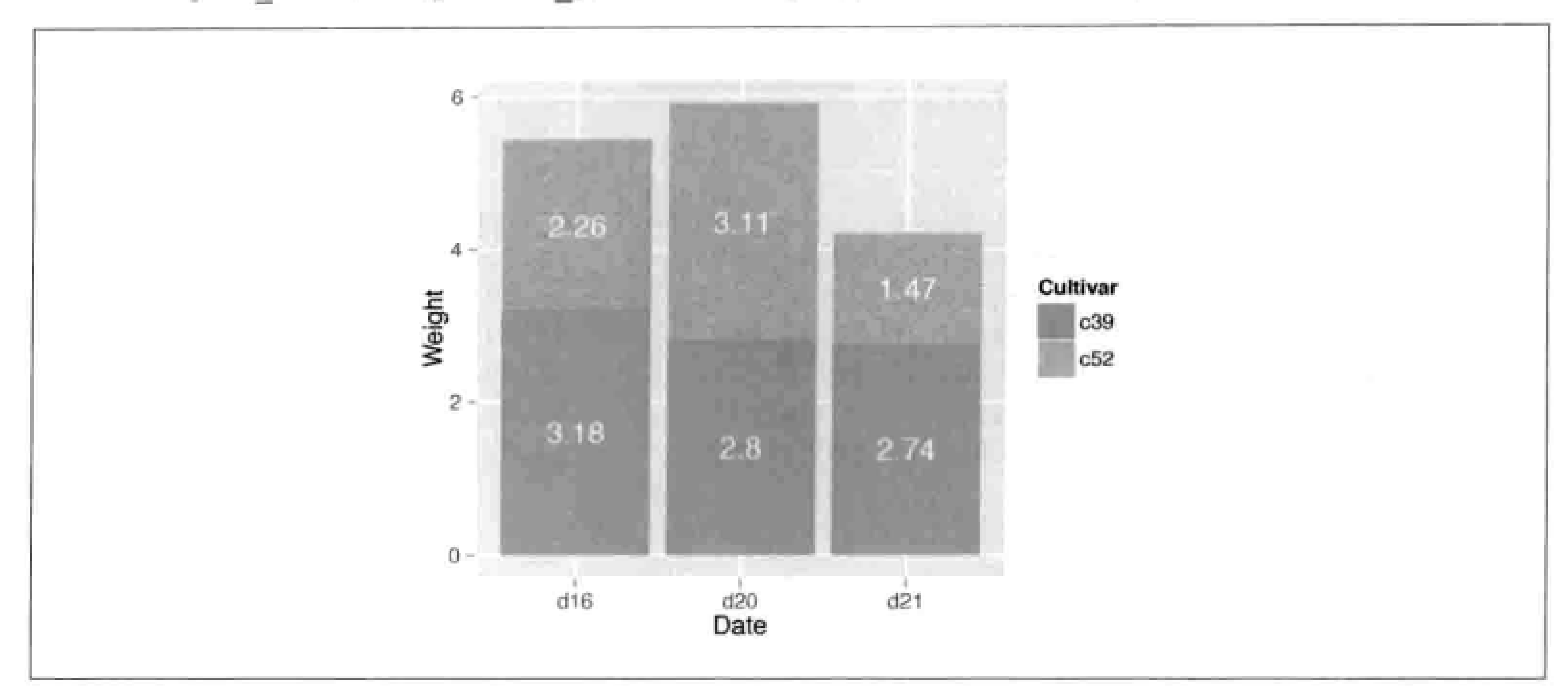


图 3-25　置于堆积条形图条形中部的数据标签

为了得到效果更好的条形图（见图 3-26），我们修改一下图例顺序和颜色，将数据标签置于条形中间，同时通过字号参数（size）缩小标签字号，并调用 paste 函数在标签后面添加“kg”，为了使得标签保留两位小数我们还需调用 format 函数：

```
ggplot(ce, aes(x=Date, y=Weight, fill=Cultivar)) +
    geom_bar(stat="identity", colour="black") +
    geom_text(aes(y=label_y, label=paste(format(Weight, nsmall=2), "kg")),
              size=4)+
    guides(fill=guide_legend(reverse=TRUE)) +
    scale_fill_brewer(palette="Pastel1")
```

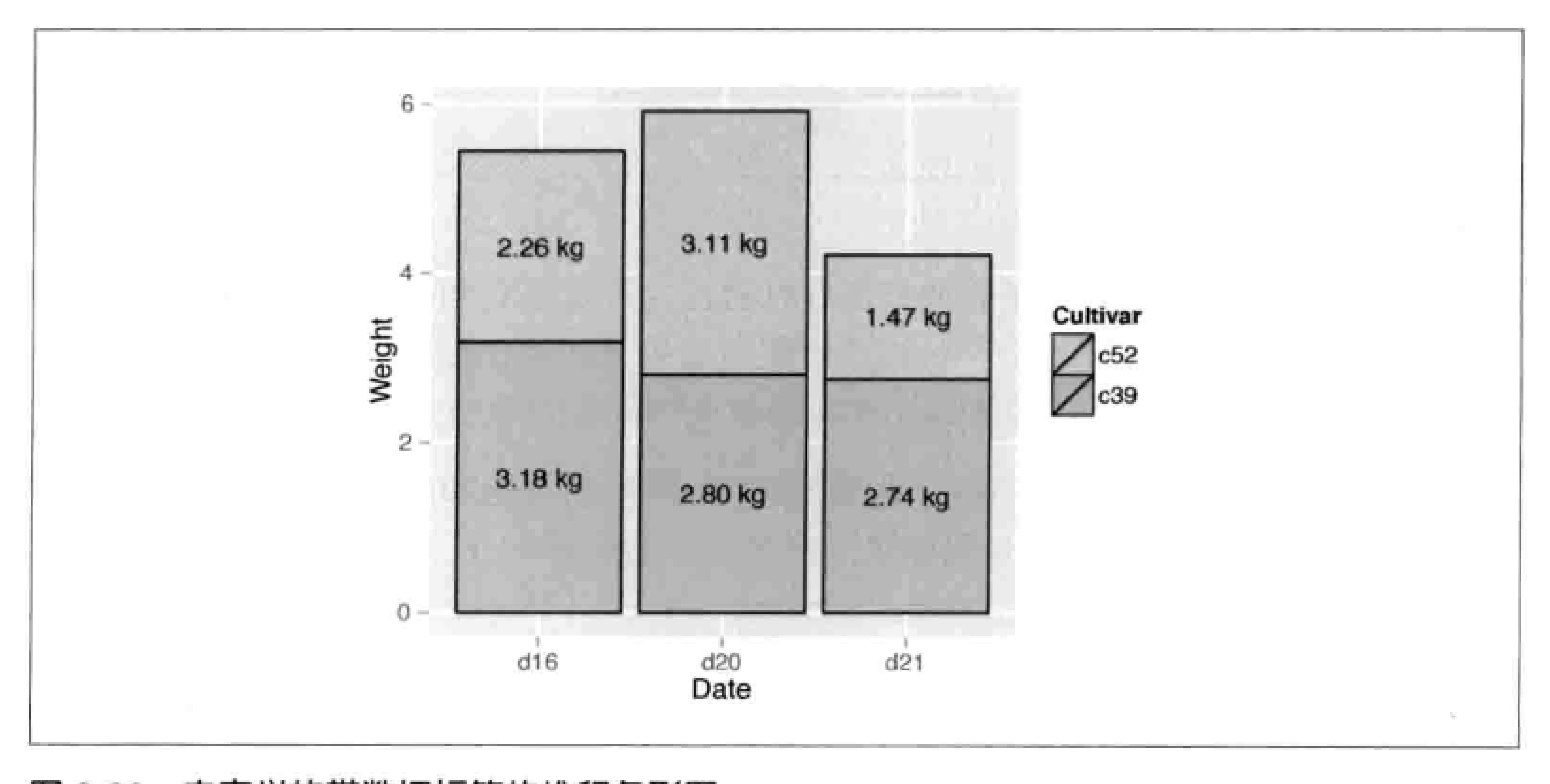


图 3-26　自定义的带数据标签的堆积条形图

另见

更多关于控制文本格式的内容可参见 9.2 节。

更多关于分组转换数据的内容可参见 15.6 节。

3.10 绘制 Cleveland 点图

问题

如何绘制 Cleveland 点图？

方法

有时人们会用 Cleverland 点图来替代条形图以减少图形造成的视觉混乱并使图形更具可读性。

最简便的绘制 Cleverland 点图的方法是直接运行 geom_point() 命令（见图 3-27）。

```
library(gcookbook) # 为了使用数据
tophit <- tophitters2001[1:25, ] # 取出 tophitters 数据集中的前 25 个数据

ggplot(tophit, aes(x=avg, y=name)) + geom_point()
```

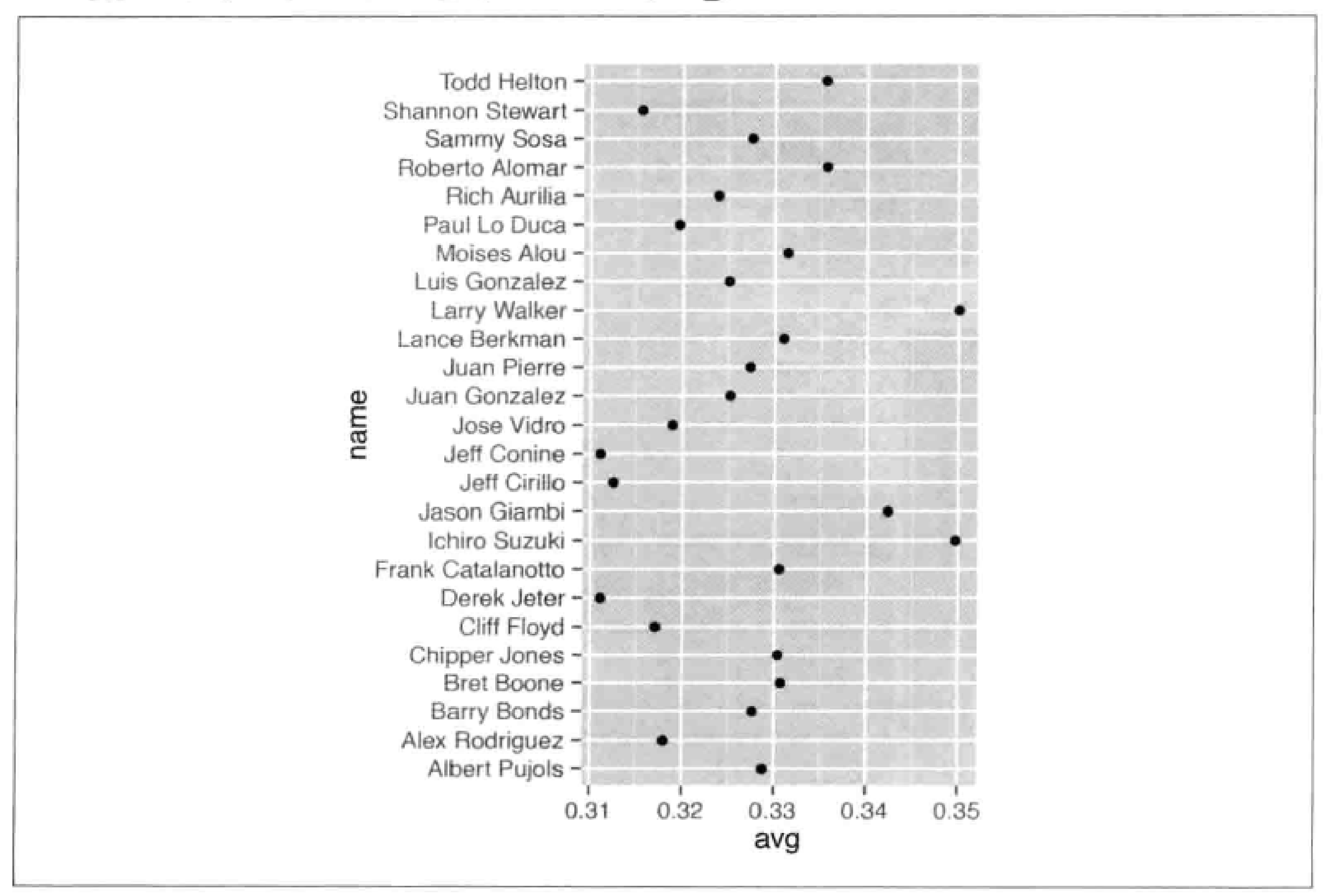


图 3-27 简单点图

讨论

tophitters2001 数据集包含很多列，这里我们只看其中三列：

```
tophit[, c("name","lg","avg")]

           name lg    avg
   Larry Walker NL 0.3501
  Ichiro Suzuki AL 0.3497
   Jason Giambi AL 0.3423
...
   Jeff Conine AL 0.3111
   Derek Jeter AL 0.3111
```

图 3-27 中的名字是按字母先后顺序排列的，这种排列方式用处不大。通常，点图中会根据 x 轴对应的连续变量的大小取值对数据进行排序。

尽管 tophit 的行顺序恰好与 avg 的大小顺序一致，但这并不意味着在图中也是这样排序的。在点图的默认设置下，坐标轴上的变量通常会根据变量类型自动选取合适的排序方式。本例中变量 name 属于字符串类型，因此，点图根据字母先后顺序对其进行了排序。当变量是因子型变量时，点图会根据定义好的因子水平顺序对其进行排序。现在，我们想根据变量 avg 对变量 name 进行排序。

我们可以借助 reorder(name,avg) 函数实现这一过程。该命令会先将 name 转化为因子，然后，根据 avg 对其进行排序。为使图形效果更好，我们借助图形主题系统（Theming System）删除垂直网格线，并将水平网格线的线型修改为虚线（见图 3-28）。

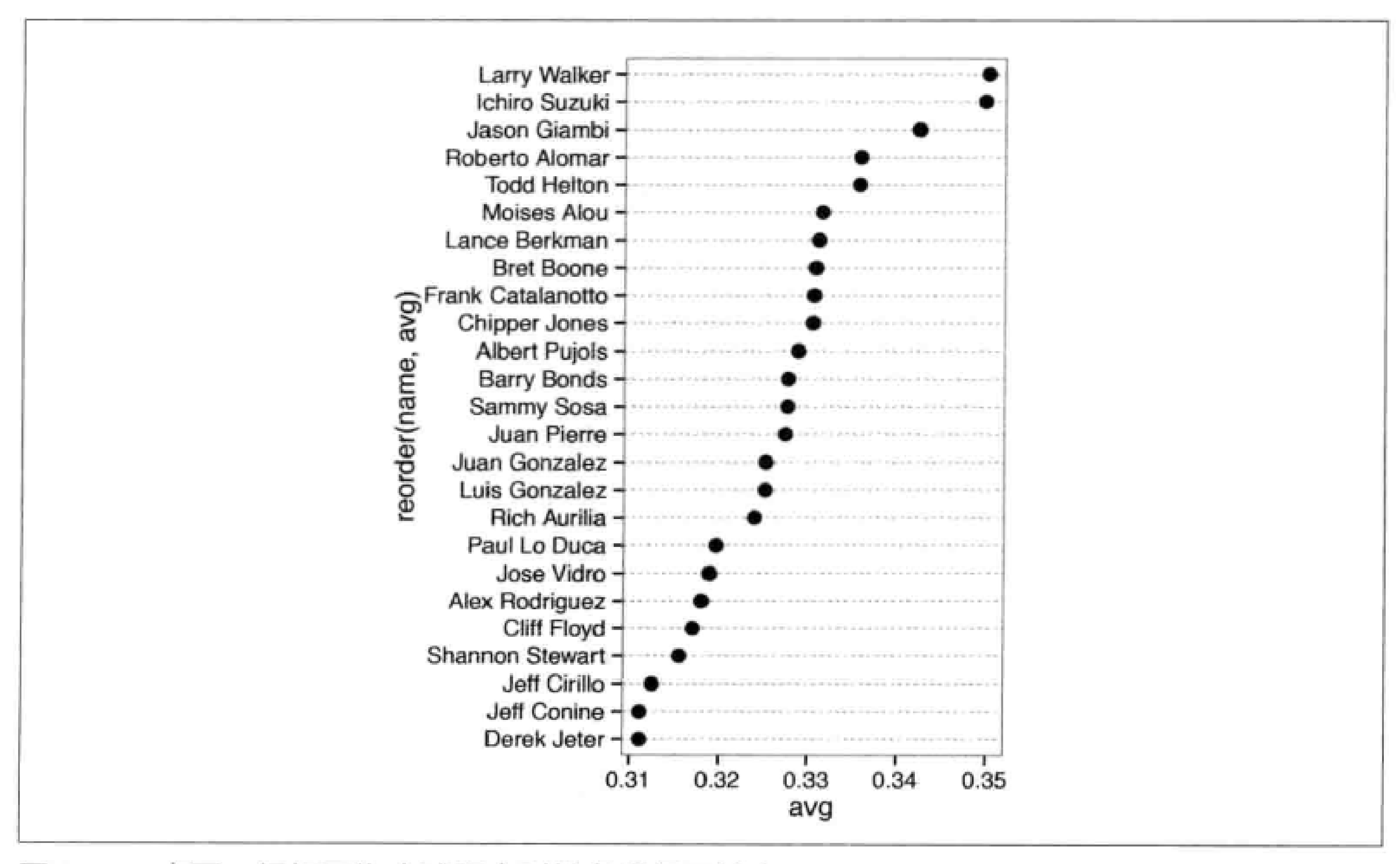


图 3-28　点图，根据平均击球距离对姓名进行了排序

```
ggplot(tophit, aes(x=avg, y=reorder(name,avg))) +
    geom_point(size=3) +      # 使用更大的点
    theme_bw() +
    theme(panel.grid.major.x = element_blank(),
          panel.grid.minor.x = element_blank(),
          panel.grid.major.y = element_line(colour="grey60", linetype="dashed"))
```

我们也可以将点图的 x 轴和 y 轴互换，互换后，x 轴对应于姓名，y 轴将对应于数值，如图 3-29 所示。我们也可以将数据标签旋转 60° 。

```
ggplot(tophit, aes(x=reorder(name, avg), y=avg)) +
    geom_point(size=3) +  # 使用更大的点
    theme_bw() +
    theme(axis.text.x = element_text(angle=60,hjust=1),
          panel.grid.major.y = element_blank(),
          panel.grid.minor.y = element_blank(),
          panel.grid.major.x = element_line(colour="grey60", linetype="dashed"))
```

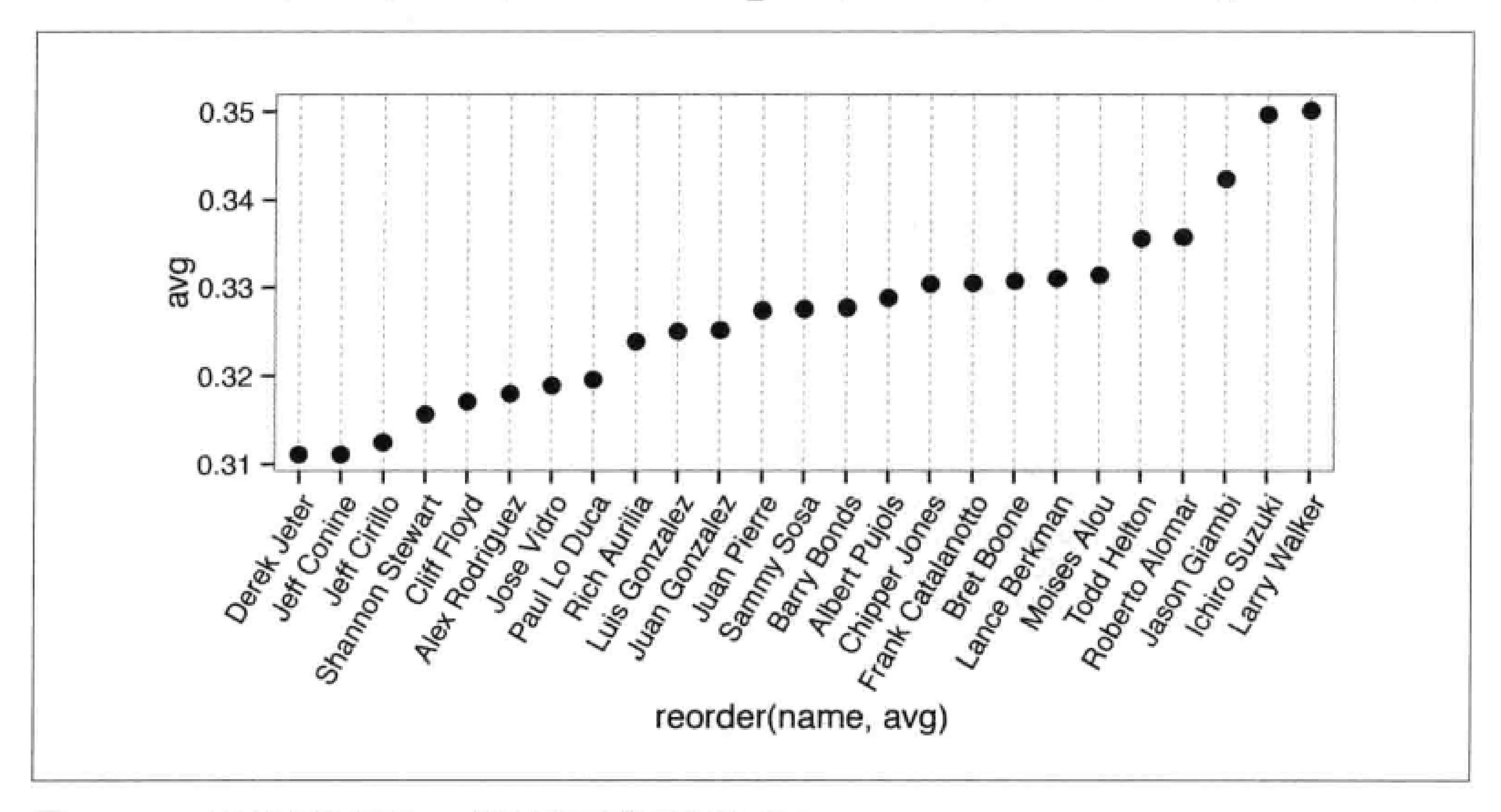


图 3-29　x 轴对应于名字，y 轴对应于变量值的点图

有时候，根据其他变量对样本进行分组很有用。这里我们根据因子 `lg` 对样本进行分组，因子 `lg` 对应有 `NL` 和 `AL` 两个水平，分别表示国家队（National League）和美国队（American league）。我们依次根据 `lg` 和 `avg` 对变量进行排序。遗憾的是，`reorder()` 函数只能根据一个变量对因子水平进行排序，所以我们只能手动实现上述过程。

```
# 提取出 name 变量，依次根据变量 lg 和 avg 对其进行排序
nameorder <- tophit$name[order(tophit$lg, tophit$avg)]

# 将 name 转化为因子，因子水平与 nameorder 一致
tophit$name <- factor(tophit$name, levels=nameorder)
```

绘制点图时（见图 3-30），我们把 `lg` 变量映射到点的颜色属性上。借助 `geom_seg-`

ment() 函数用"以数据点为端点的线段"代替贯通全图的网格线。注意 geom_segment() 函数需要设定 x、y、xend 和 yend 四个参数：

```
ggplot(tophit, aes(x=avg, y=name)) +
    geom_segment(aes(yend=name), xend=0, colour="grey50") +
    geom_point(size=3, aes(colour=lg)) +
    scale_colour_brewer(palette="Set1", limits=c("NL","AL")) +
    theme_bw() +
    theme(panel.grid.major.y = element_blank(),     # 删除水平网格线
          legend.position=c(1, 0.55),               # 将图例放置在绘图区域中
          legend.justification=c(1, 0.5))
```

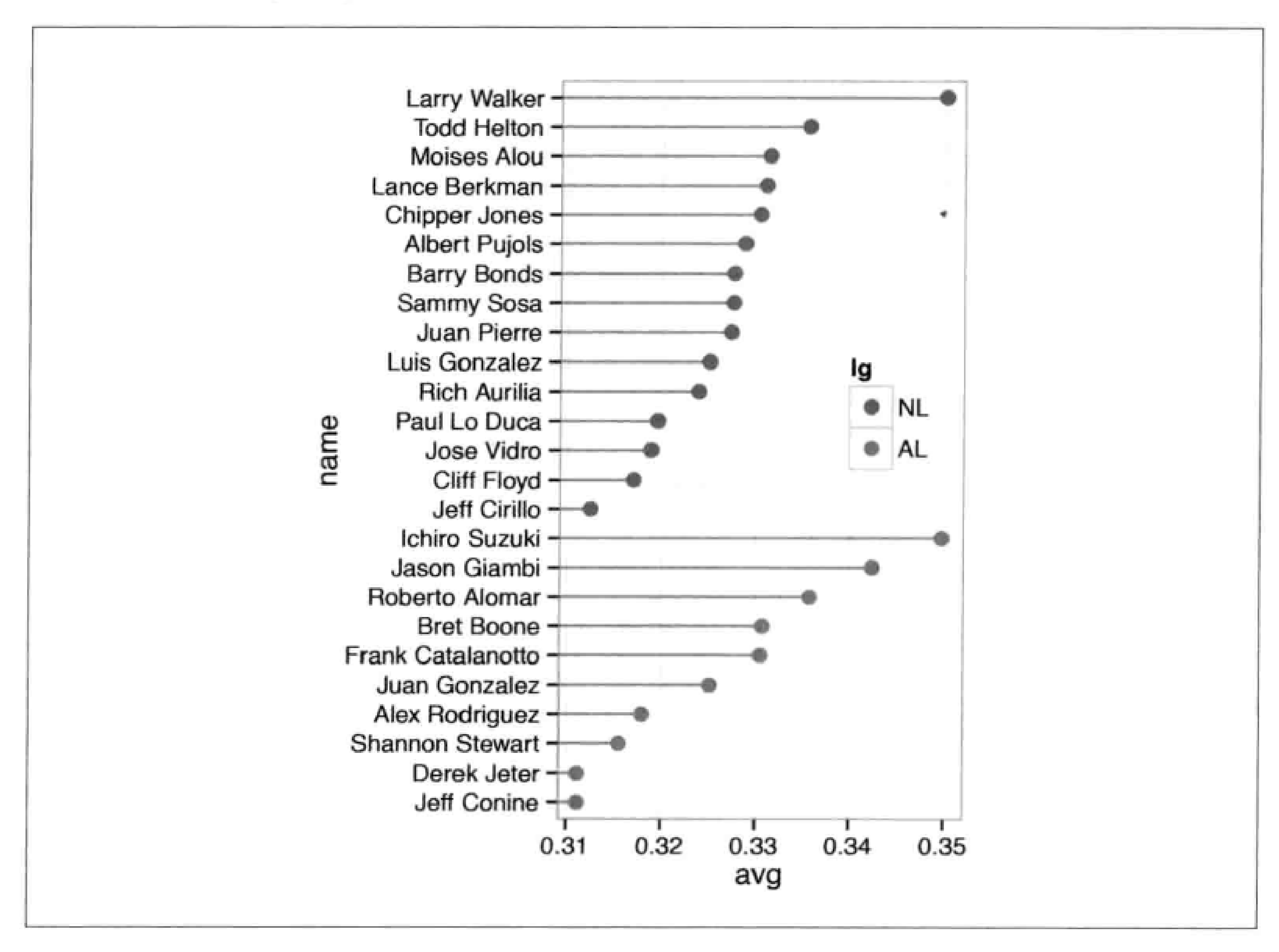


图 3-30　以队为分组变量的火柴杆图

另外一种分组展示数据的方式是分面，如图 3-31 所示。分面条形图中的条形的堆叠顺序与图 3-30 中的堆叠顺序有所不同；要修改分面显示的堆叠顺序只有通过调整 lg 变量的因子水平来实现。

```
ggplot(tophit, aes(x=avg, y=name)) +
    geom_segment(aes(yend=name), xend=0, colour="grey50") +
    geom_point(size=3, aes(colour=lg)) +
    scale_colour_brewer(palette="Set1", limits=c("NL","AL"), guide=FALSE) +
    theme_bw() +
```

```
theme(panel.grid.major.y = element_blank()) +
facet_grid(lg ~ ., scales="free_y", space="free_y")
```

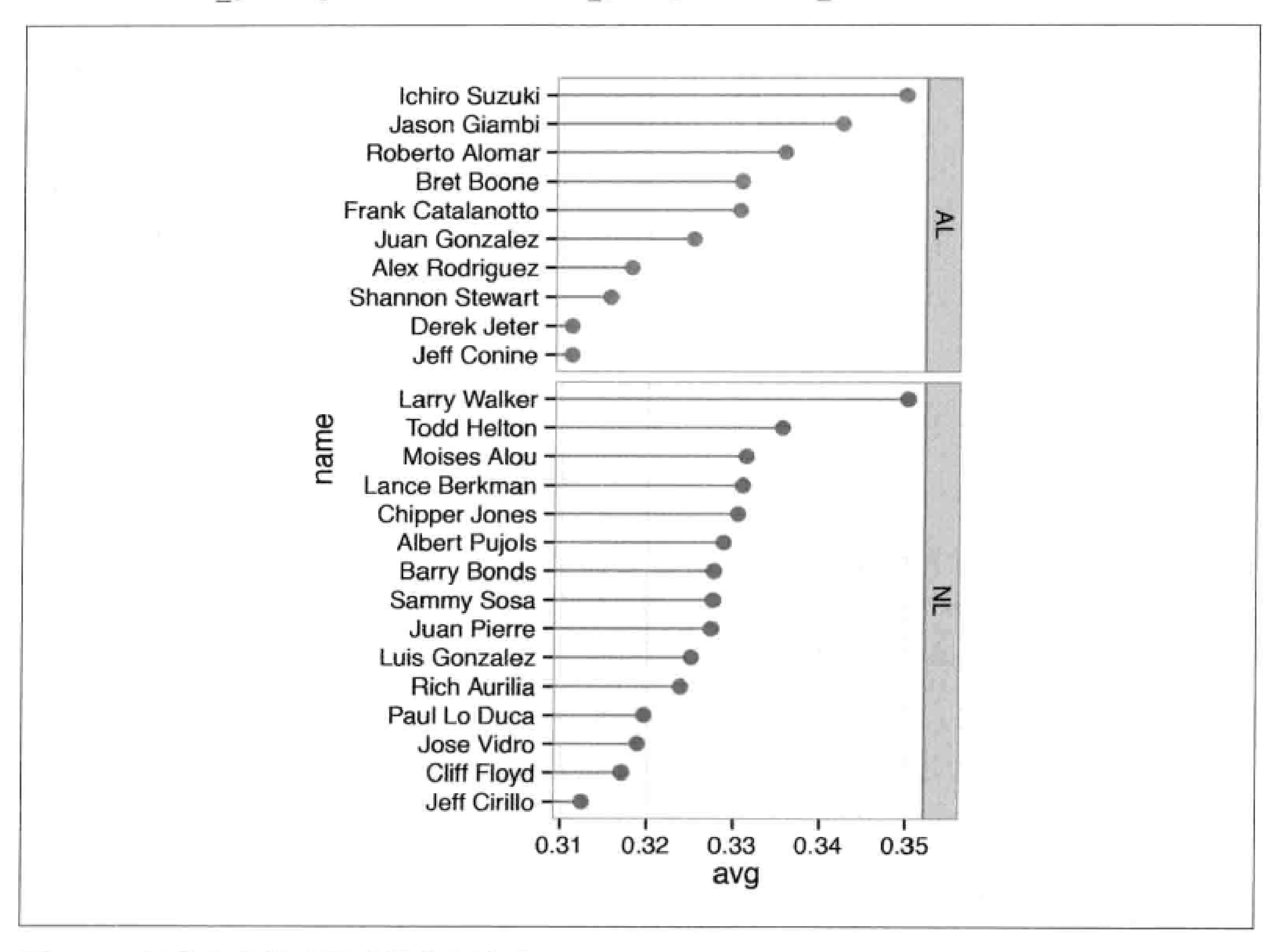


图 3-31　以队为分组变量进行分面绘图

另见

更多关于调整因子水平顺序的内容可参见 15.8 节。关于基于其他变量调整因子水平顺序的细节内容可参见 15.9 节。

更多关于调整图例位置的内容可参见 10.2 节。更多关于隐藏网格线的内容，可参见 9.6 节。

第4章

折线图

折线图通常用来对两个连续变量之间的相互依存关系进行可视化，其中，x 轴对应于自变量，y 轴对应于因变量。一般来说，折线图的 x 轴对应的是时间变量，但也可以用来表示诸如实验对象的药剂量等连续型变量。

当然，跟条形图类似，折线图的用法也有例外。有时候，折线图的 x 轴也可以与离散型变量相对应，但此时只适用于变量为有序离散型变量（比如“小”、“中”、“大”）的情形，而不适用于无序变量（比如“牛”、“鹅”、“猪”等）。本章中的大部分案例用到的都是连续型变量 x，在其中一个案例中，我们会将连续型变量转化为因子型变量，因此，它也可以看作是一个针对离散型变量绘制折线图的例子。

4.1 绘制简单折线图

问题

如何绘制简单折线图？

方法

运行 ggplot() 和 geom_line() 函数，并分别指定一个变量映射给 x 和 y（见图 4-1）。

```
ggplot(BOD, aes(x=Time, y=demand)) + geom_line()
```

讨论

对于这个简单的数据框，x 对应的变量 Time 和 y 对应的变量 demand 分别对应于数据框的两列数据：

```
BOD
Time demand
```

```
1   8.3
2  10.3
3  19.0
4  16.0
5  15.6
7  19.8
```

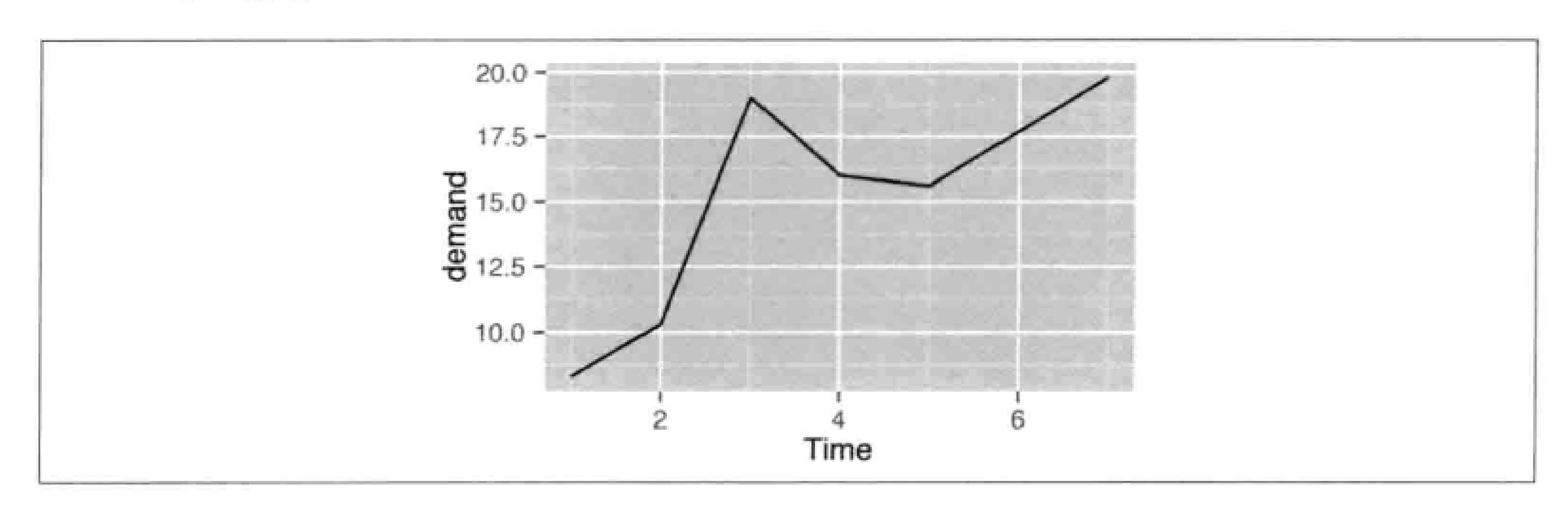


图 4-1　简单折线图

折线图的 *x* 轴既可以对应于离散型（分类）变量，也可以对应于连续型（数值型）变量。本例中 Time 变量为连续型变量，但我们可以借助 factor() 函数将其转化为因子型变量，然后，将其当作分类变量来处理（见图 4-2）。当 x 对应于因子型变量时，必须使用命令 aes(group=1) 以确保 ggplot() 知道这些数据点属于同一个分组，从而应该用一条折线连在一起（关于为什么因子型变量必须设定 group 的内容可以参见 4.3 节）。

```
BOD1 <- BOD # 将数据复制一份
BOD1$Time <- factor(BOD1$Time)
ggplot(BOD1, aes(x=Time, y=demand, group=1)) + geom_line()
```

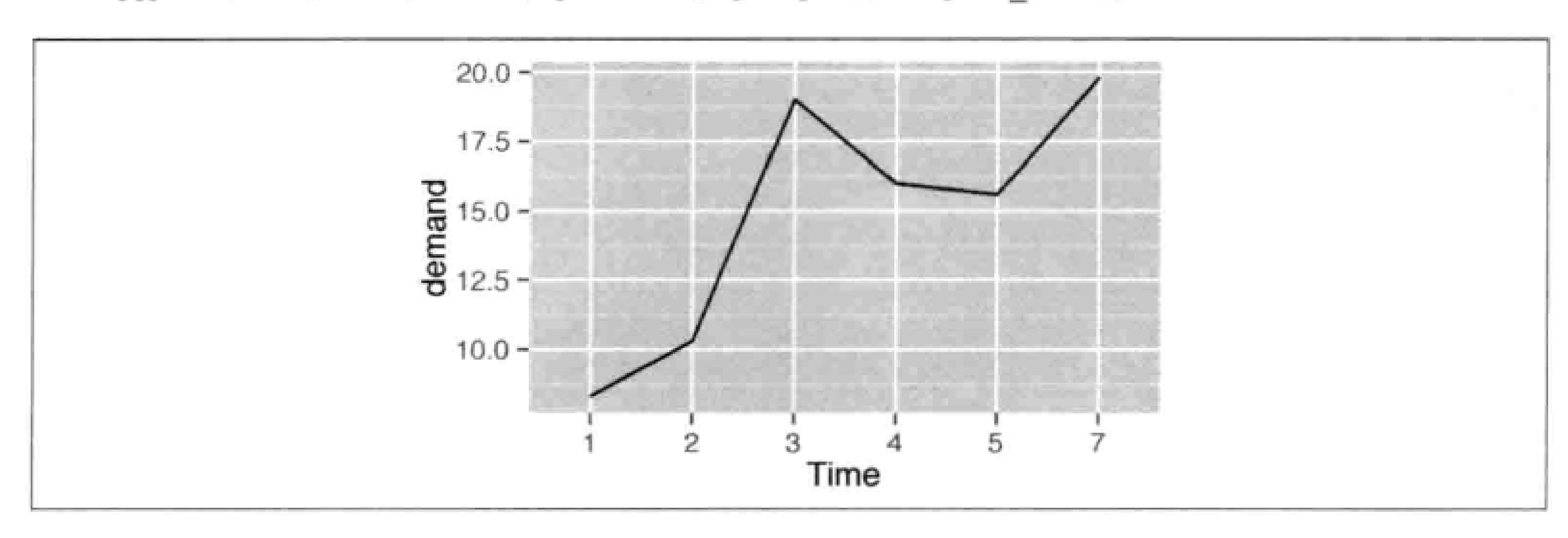


图 4-2　*x* 轴对应于分类变量的简单折线图（注意 *x* 轴上没有对应于水平 6 的取值）

数据集 BOD 中没有对应于 Time=6 的数据点，因此当 Time 被转化为因子型变量时，它并没有 6 这个水平。因子型变量对应于分类值，这里的 6 只是其中一个可能的取值。因为数据集中恰好没有对应于该水平的数据点，所以，*x* 轴上没有绘制相应的取值。

默认情况下，ggplot2 绘制的折线图的 *y* 轴范围刚好能容纳数据集中的 *y* 值。对于某些数据而言，我们将 *y* 轴的起点设定为 0 点会更合适。你可以运行 ylim() 设定 *y* 轴范围

或者运行含一个参数的 expand_limit() 扩展 y 轴的范围。下面的命令将 y 轴的范围设定为 0 到 BOD 中 demand 变量的最大值。

```
# 运行下面的命令得到的结果是相同的
ggplot(BOD, aes(x=Time, y=demand)) + geom_line() + ylim(0, max(BOD$demand))
ggplot(BOD, aes(x=Time, y=demand)) + geom_line() + expand_limits(y=0)
```

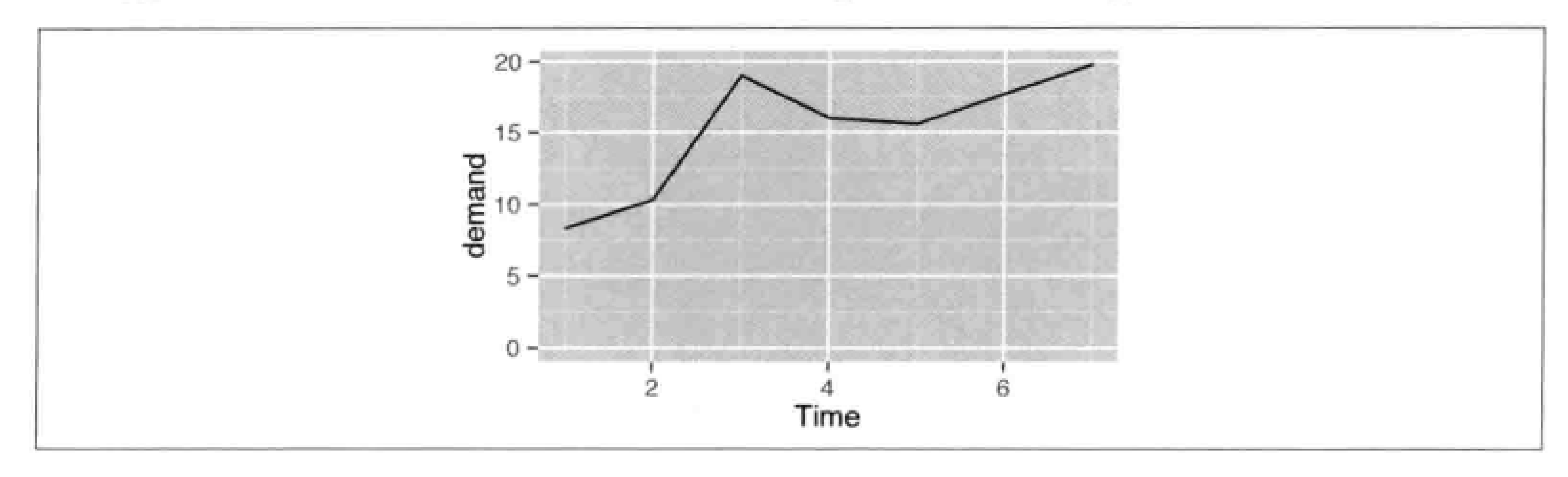


图 4-3　手动设定 y 轴范围的折线图

另见

更多关于控制坐标轴范围的内容参见 8.2 节。

4.2　向折线图添加数据标记

问题

如何向折线图添加数据标记？

方法

在代码中加上 geom_point()（见图 4-4）：

```
ggplot(BOD, aes(x=Time, y=demand)) + geom_line() + geom_point()
```

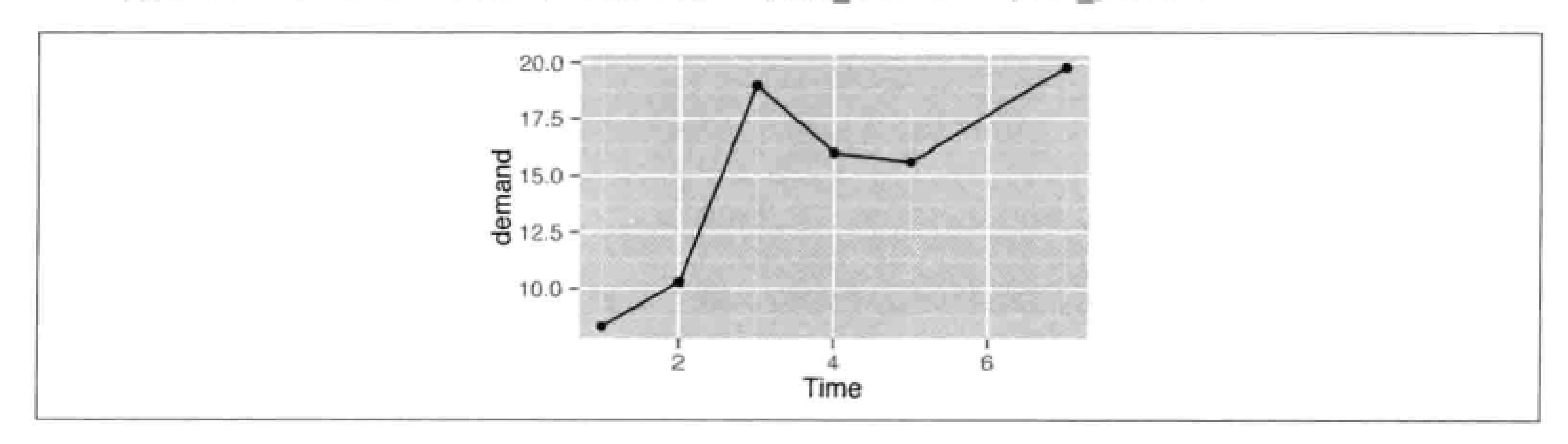


图 4-4　添加了数据标记的折线图

讨论

有时候，在折线图上添加数据标记很有用处。当数据点的密度较低或者数据采集频率

不规则时尤其有用。比如，BOD 数据集中没有与 Time=6 相对应的输入，然而，这在单独的一张折线图看起来并不明显（可比较一下图 4-3 和图 4-4）。

worldpop 数据集对应的采集时间间隔不是常数。时间距今较久远的数据采集频率比新近不久的数据采集频率低。折线图中的数据标记表明了数据的采集时间（见图 4-5）：

```
library(gcookbook) # 为了使用数据

ggplot(worldpop, aes(x=Year, y=Population)) + geom_line() + geom_point()

# 当 y 轴取对数时也一样
ggplot(worldpop, aes(x=Year, y=Population)) + geom_line() + geom_point() +
    scale_y_log10()
```

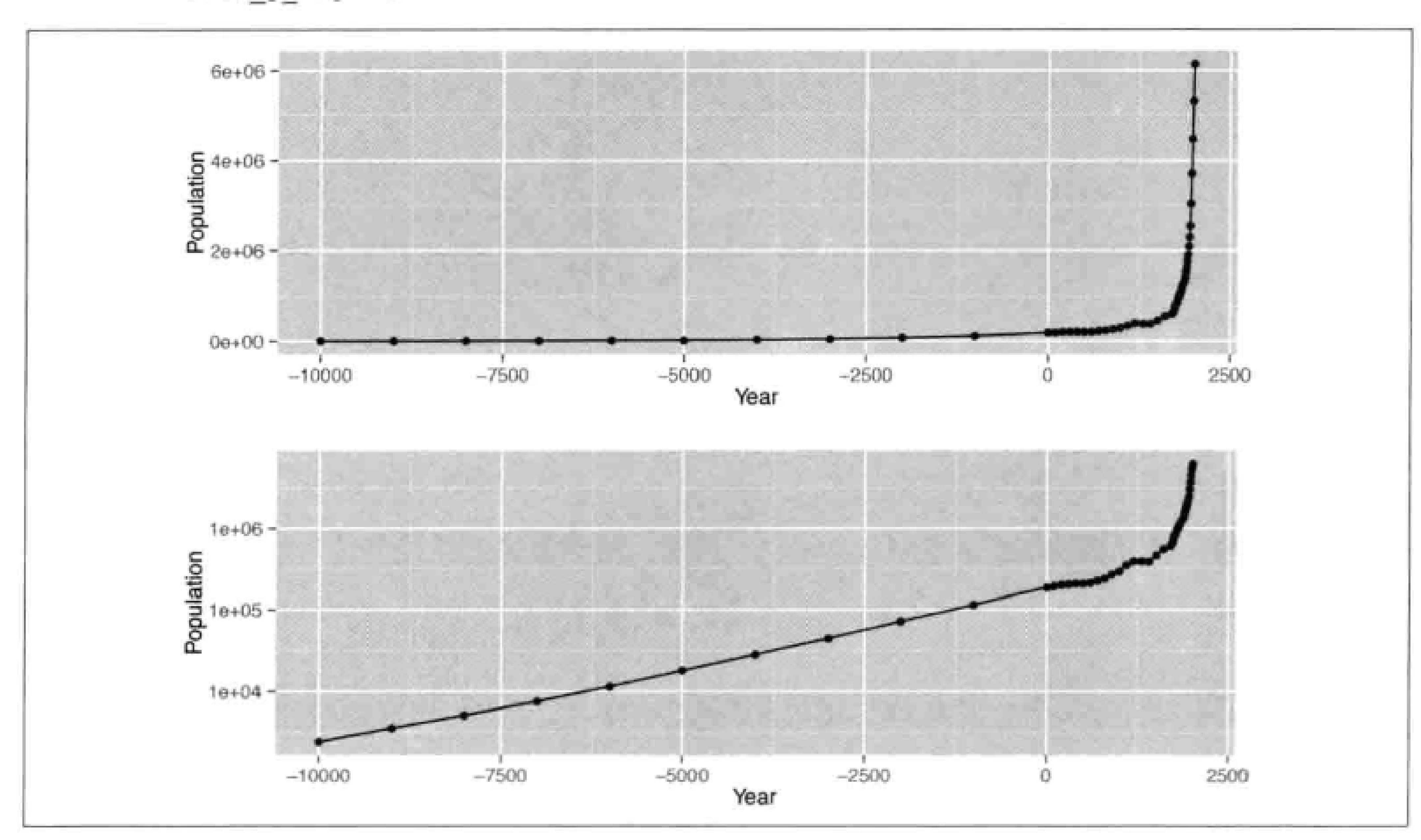


图 4-5　上图：数据标记表明了数据的采集时间　下图：对 *y* 轴取对数

从对 *y* 轴取对数的折线图上可以看出：在过去数千年中人口增长率有所增加。公元元年之前的人口增长率接近常数，约每 5000 年增加 10 倍。从图中也可以看出，近年来的人口普查频率比以往更为频繁，数据也更为准确。

另见

更多关于修改数据标记样式的内容可参见 4.5 节。

4.3　绘制多重折线图

问题

如何绘制多重折线图？

方法

在分别设定一个映射给 x 和 y 的基础上，再将另外一个（离散型）变量映射给颜色（colour）或者线型（linetype）即可，如图 4-6 所示。

```
# 载入plyr包，便于我们使用ddply()函数创建样本数据集
library(plyr)
# 对ToothGrowth数据集进行汇总
tg <- ddply(ToothGrowth, c("supp", "dose"), summarise, length=mean(len))

# 将supp映射给颜色(colour)
ggplot(tg, aes(x=dose, y=length, colour=supp)) + geom_line()

# 将supp映射给线型(linetype)
ggplot(tg, aes(x=dose, y=length, linetype=supp)) + geom_line()
```

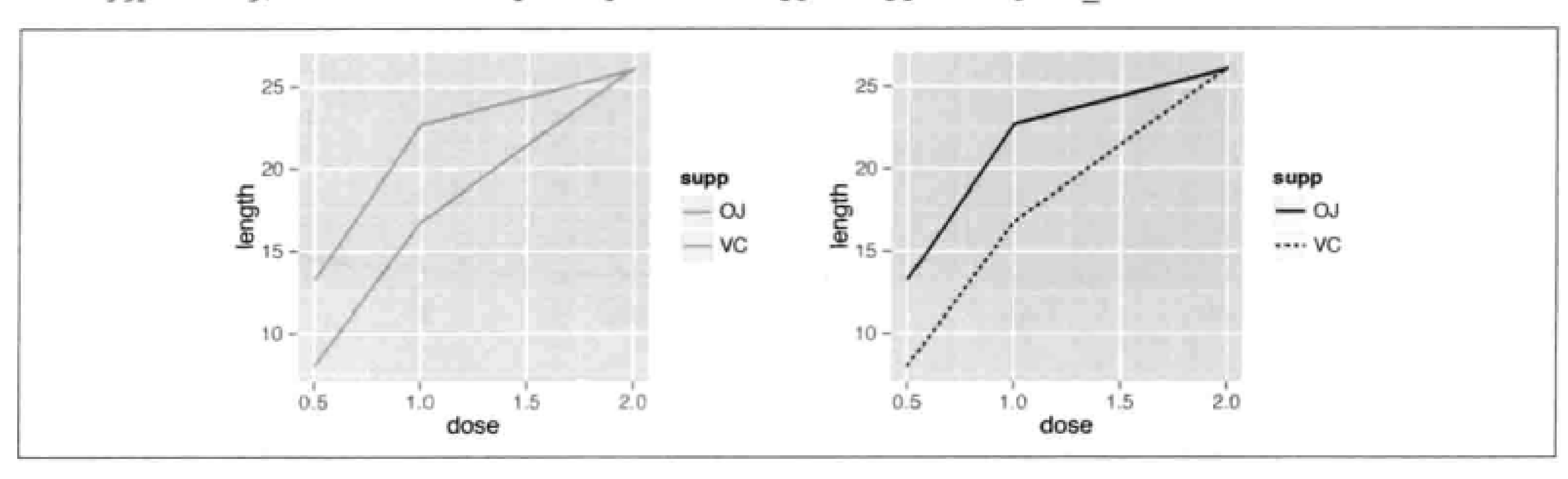


图 4-6　左图：将变量映射给颜色（colour） 右图：将变量映射给线型（linetype）

讨论

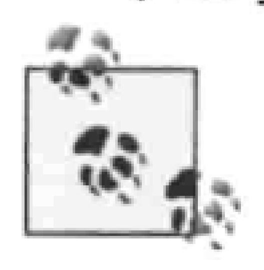
tg 数据集共有三列，其中一列是我们映射给颜色（colour）和线型（linetype）的 supp 变量：

```
tg

 supp dose length
   OJ  0.5  13.23
   OJ  1.0  22.70
   OJ  2.0  26.06
   VC  0.5   7.98
   VC  1.0  16.77
   VC  2.0  26.14

str(tg)

'data.frame':   6 obs. of  3 variables:
 $ supp  : Factor w/ 2 levels "OJ","VC": 1 1 1 2 2 2
 $ dose  : num  0.5 1 2 0.5 1 2
 $ length: num  13.23 22.7 26.06 7.98 16.77 ...
```



如果 *x* 变量是因子，你必须同时告诉 ggplot() 用来分组的变量，正如接下来要介绍的那样。

折线图的 *x* 轴既可以对应于连续型变量也可以对应于离散型变量。有时候，映射给 x 的变量虽然被存储为数值型变量，但被看作分类变量来处理。本例中，dose 变量有三个取值：0.5、1.0 和 2。或许你更想将其当作分类变量而不是连续型变量来处理，那么运行 factor() 函数将其转化为因子（如图 4-7 所示）。



```
ggplot(tg, aes(x=factor(dose), y=length, colour=supp, group=supp)) + geom_line()
```

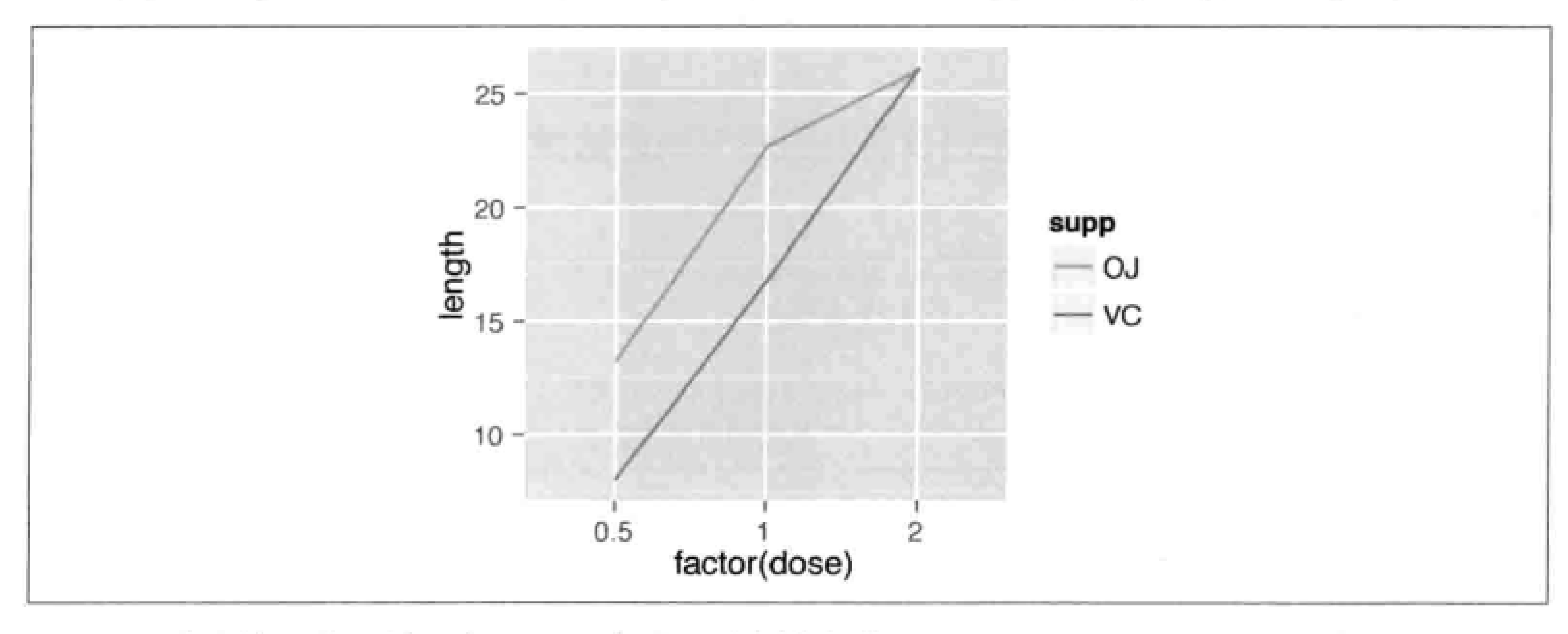


图 4-7　将连续型变量转化为因子型变量后绘制的折线图

注意，不可缺少 group=supp 语句，否则，ggplot() 会不知如何将数据组合在一起绘制折线图，从而会报错：

```
ggplot(tg, aes(x=factor(dose), y=length, colour=supp)) + geom_line()
```

geom_path: Each group consists of only one observation. Do you need to adjust the group aesthetic?

当分组不正确时会遇见的另一种问题是，折线图会变成锯齿状，如图 4-8 所示。

```
ggplot(tg, aes(x=dose, y=length)) + geom_line()
```

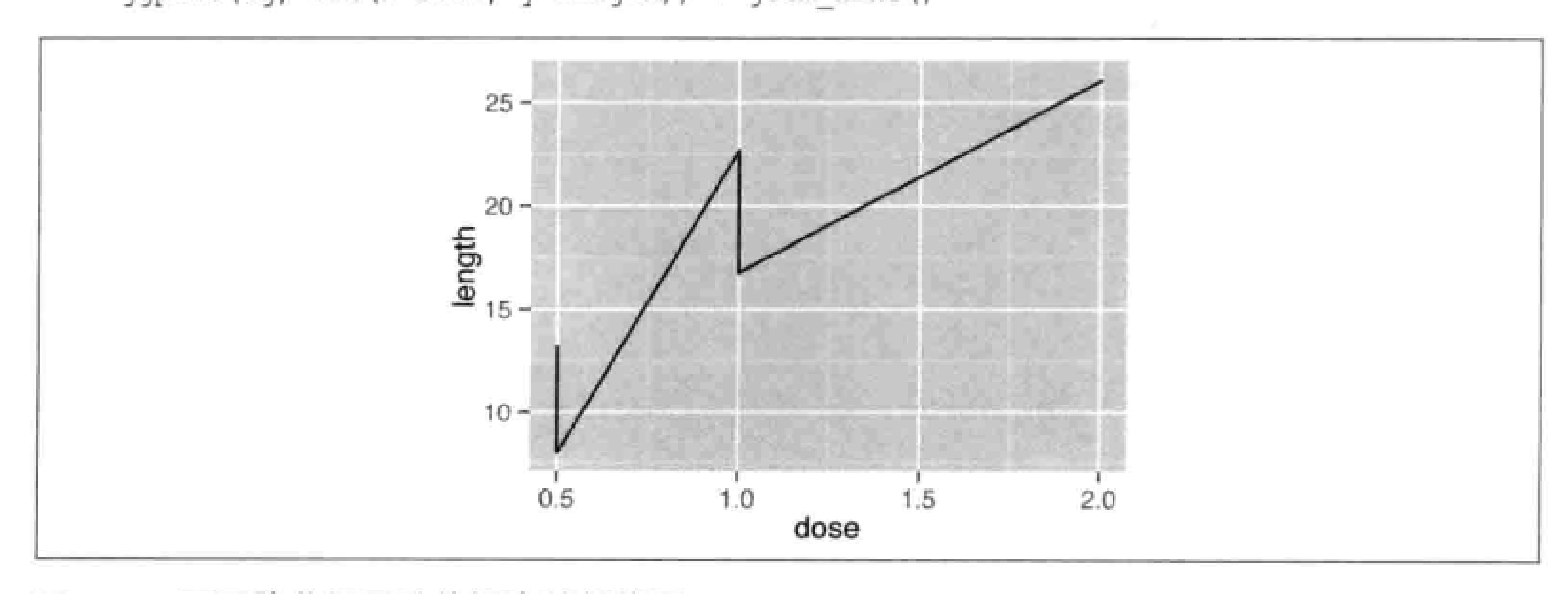


图 4-8　不正确分组导致的锯齿状折线图

导致这种情况的原因在于 *x* 在每个位置都对应于多个点，ggplot() 误以为这些点属于

同一组数据而将其用一根折线相连，结果形成了锯齿状折线图。如果将任意离散型变量映射给 colour 或者 linetype，ggplot() 会以其为分组变量对数据进行分组。如果你想借助其他变量对数据进行分组（未映射给图形属性）则需使用 group。



有疑问时，或者如果你的折线图看起来不太合理，可以试着用 group 明确指定分组变量。这种问题十分常见，因为 ggplot() 不知道如何对折线图数据进行分组。

如果折线图上有数据标记，你也可以将分组变量映射给数据标记的属性，诸如 shape 和 fill 等（见图 4-9）。

```
ggplot(tg, aes(x=dose, y=length, shape=supp)) + geom_line() +
    geom_point(size=4)            # 更大的点

ggplot(tg, aes(x=dose, y=length, fill=supp)) + geom_line() +
    geom_point(size=4, shape=21)      # 使用有填充色的点
```

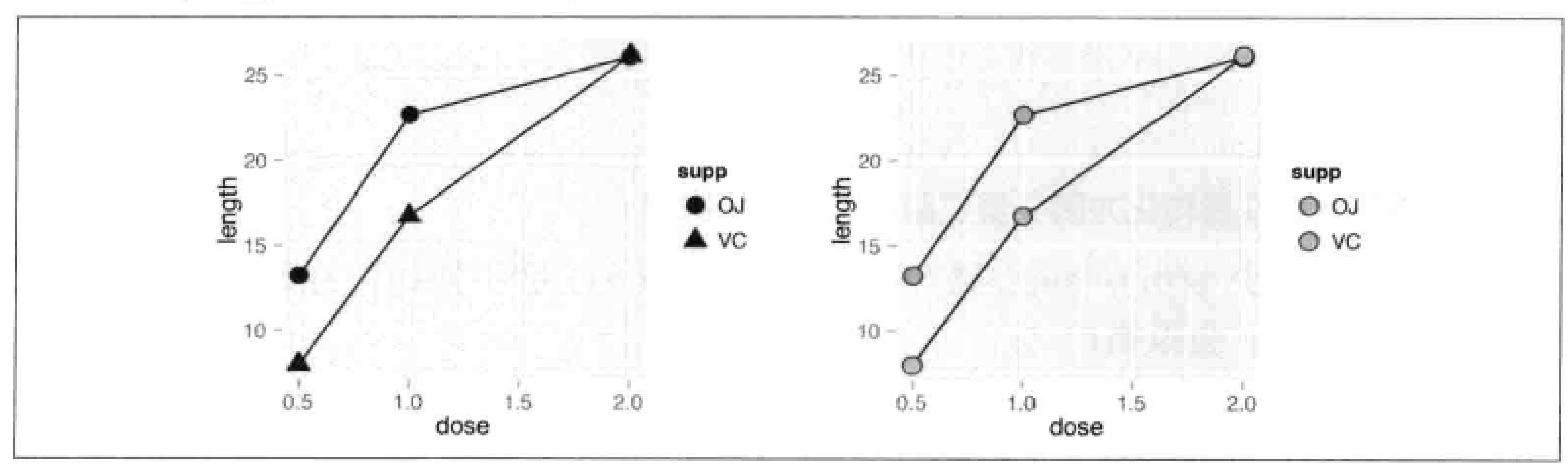


图 4-9　左图：对应于不同点形的折线图　右图：对应于不同颜色的折线图

有时，数据标记会相互重叠。我们需要令其彼此错开。这意味着要将它们的位置左移或者右移（见图 4-10）。同时，需要相应地左移或者右移连接线以避免点线偏离。在这一过程中，必须指定数据标记的移动距离。

```
ggplot(tg, aes(x=dose, y=length, shape=supp)) +
    geom_line(position=position_dodge(0.2)) +           # 将连接线左右移动 0.2
    geom_point(position=position_dodge(0.2), size=4)    # 将点的位置左右移动 0.2
```

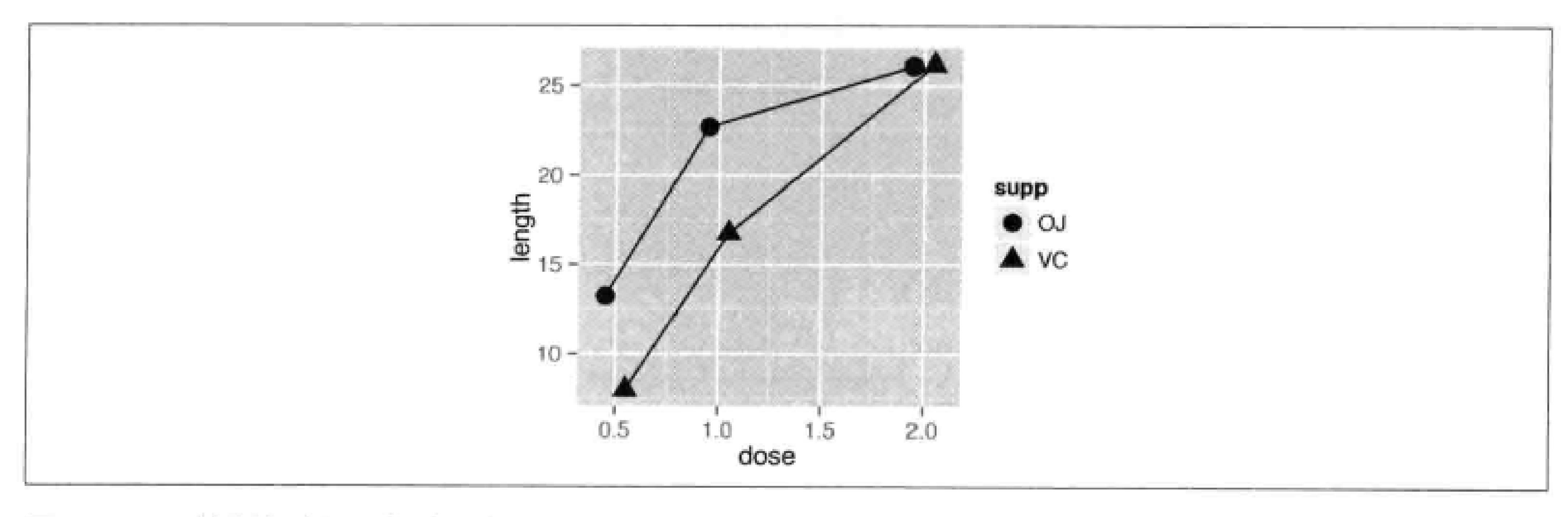


图 4-10　使数据标记点彼此错开以避免重叠

4.4 修改线条样式

问题

如何修改折线图的线条样式？

方法

通过设置线型（linetype）、线宽（size）和颜色（colour）参数可以分别修改折线的线型、线宽和颜色。

通过将这些参数的值传递给 geom_line() 函数可以设置折线图的对应属性，如图 4-11 所示。

```
ggplot(BOD, aes(x=Time, y=demand))+
    geom_line(linetype="dashed", size=1, colour="blue")
```

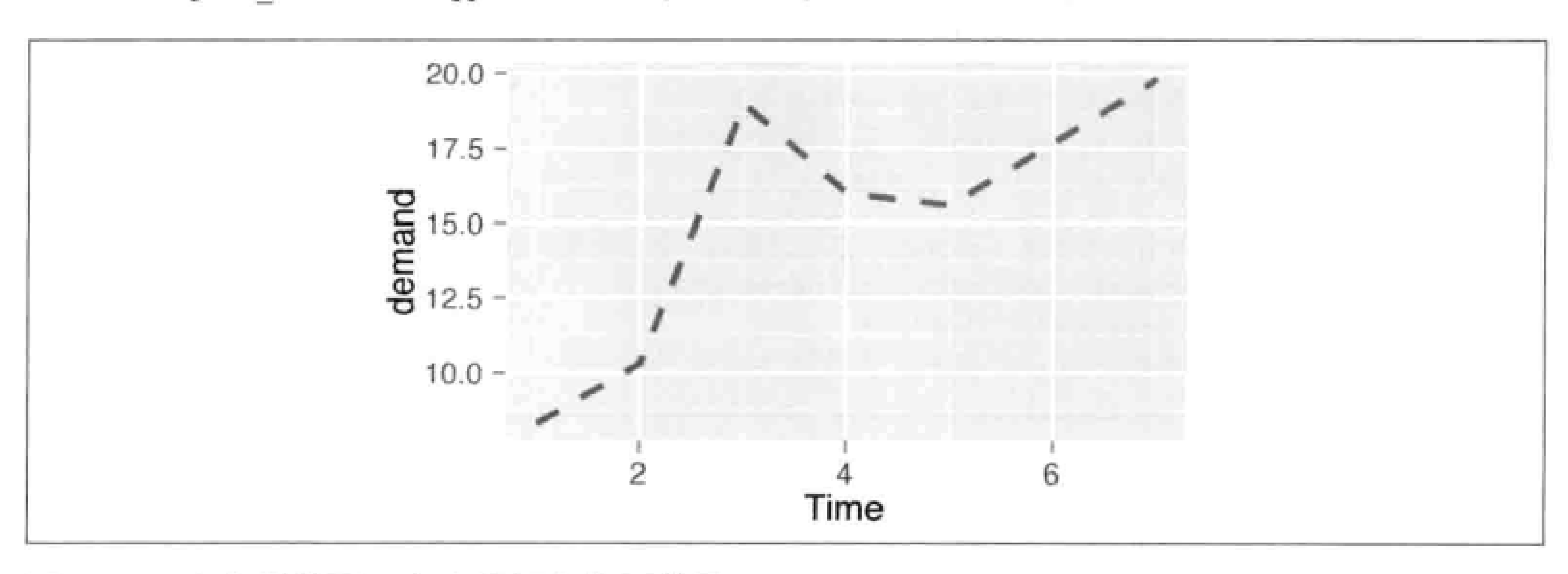


图 4-11 自定义线型、宽度和颜色的折线图

对于多重折线图而言，设定图形属性会对图上的所有折线产生影响。而将变量映射给图形属性则会使图上的折线具有不同的外观，参见 4.3 节。折线图的默认颜色并不是很吸引眼球，所以，我们可能希望使用其他调色板为图形着色，可以调用 scale_colour_brewer() 和 scale_colour_manual() 函数完成上述操作，如图 4-12 所示。

```
# 加载 plyr 包，便于调用 ddply() 函数创建例子所需的数据集
library(plyr)
# 对 ToothGrowth 数据集进行汇总
tg <- ddply(ToothGrowth, c("supp","dose"), summarise, length=mean(len))

ggplot(tg, aes(x=dose, y=length, colour=supp)) +
    geom_line() +
    scale_colour_brewer(palette="Set1")
```

讨论

在 aes() 函数外部设定颜色（colour）会将所有折线设定为同样的颜色。其他图形属性诸如线宽（size）、线型（linetype）和点形（shape）与此类似，如图 4-13 所示。

操作过程中可能需要指定分组变量。

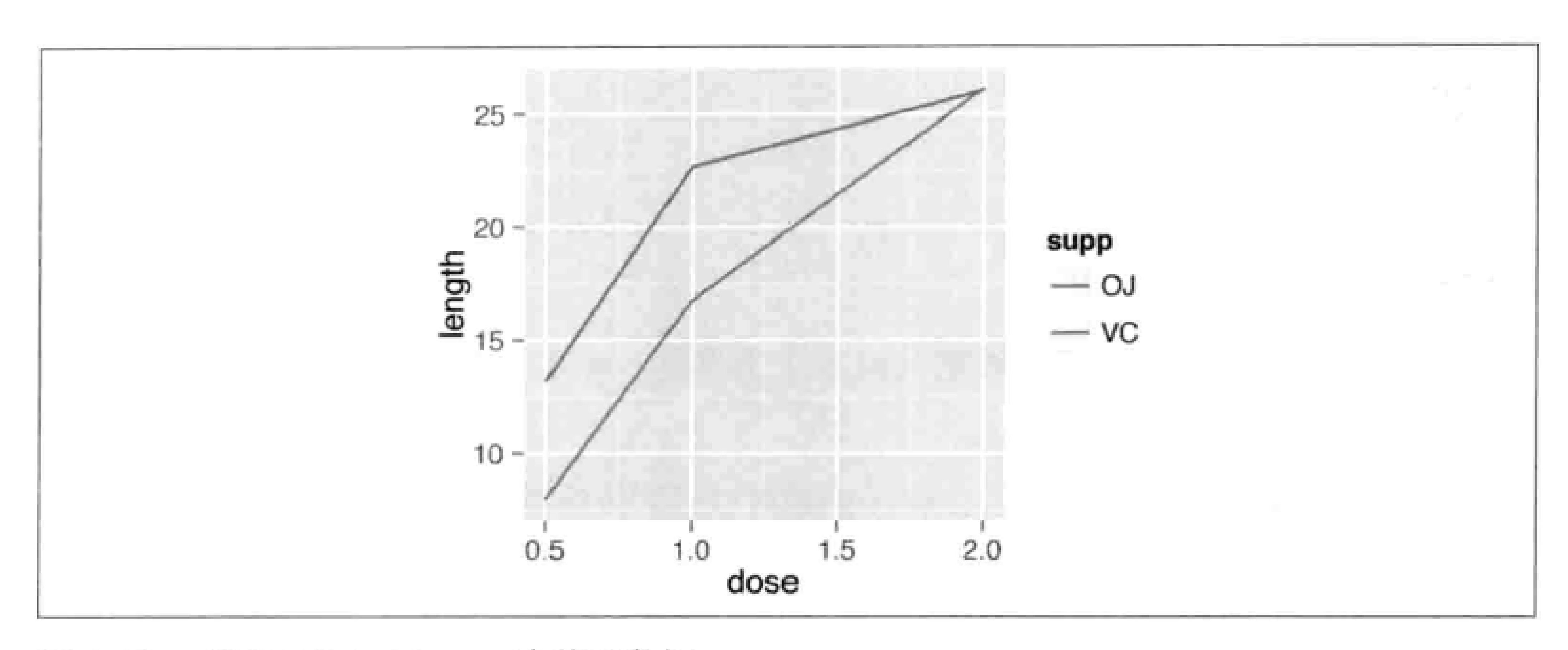


图 4-12　调用 RColorBrewer 中的调色板

```
# 如果两条折线的图形属性相同，需要指定一个分组变量
ggplot(tg, aes(x=dose, y=length, group=supp)) +
  geom_line(colour="darkgreen", size=1.5)

# 因为变量 supp 被映射给了颜色（colour）属性，所以，它自动作为分组变量
ggplot(tg, aes(x=dose, y=length, colour=supp)) +
     geom_line(linetype="dashed") +
     geom_point(shape=22, size=3, fill="white")
```

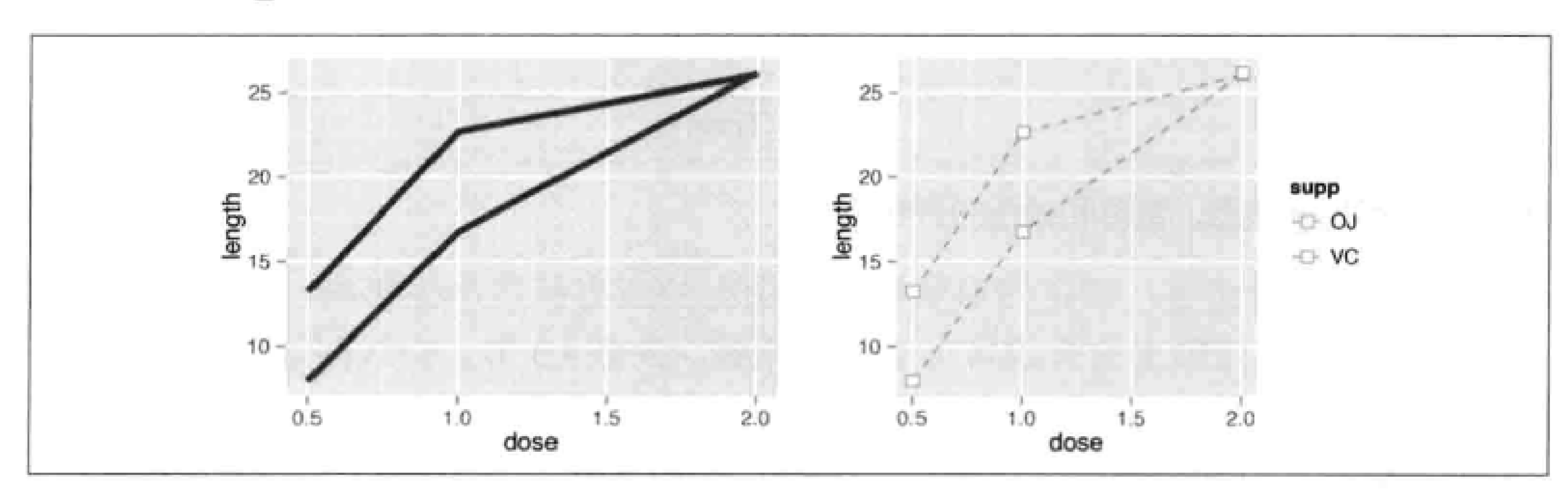


图 4-13　左图：单一颜色和宽度的折线图　右图：将变量 supp 映射给 colour 并添加数据标记的折线图

另见

更多关于使用颜色的内容可参见第 12 章。

4.5　修改数据标记样式

问题

如何修改数据标记的样式？

方法

在函数 aes() 外部设定函数 geom_point() 的大小（size）、颜色（colour）和填充色（fill）即可，如图 4-14 所示。

```
ggplot(BOD, aes(x=Time, y=demand)) +
    geom_line() +
    geom_point(size=4, shape=22, colour="darkred", fill="pink")
```

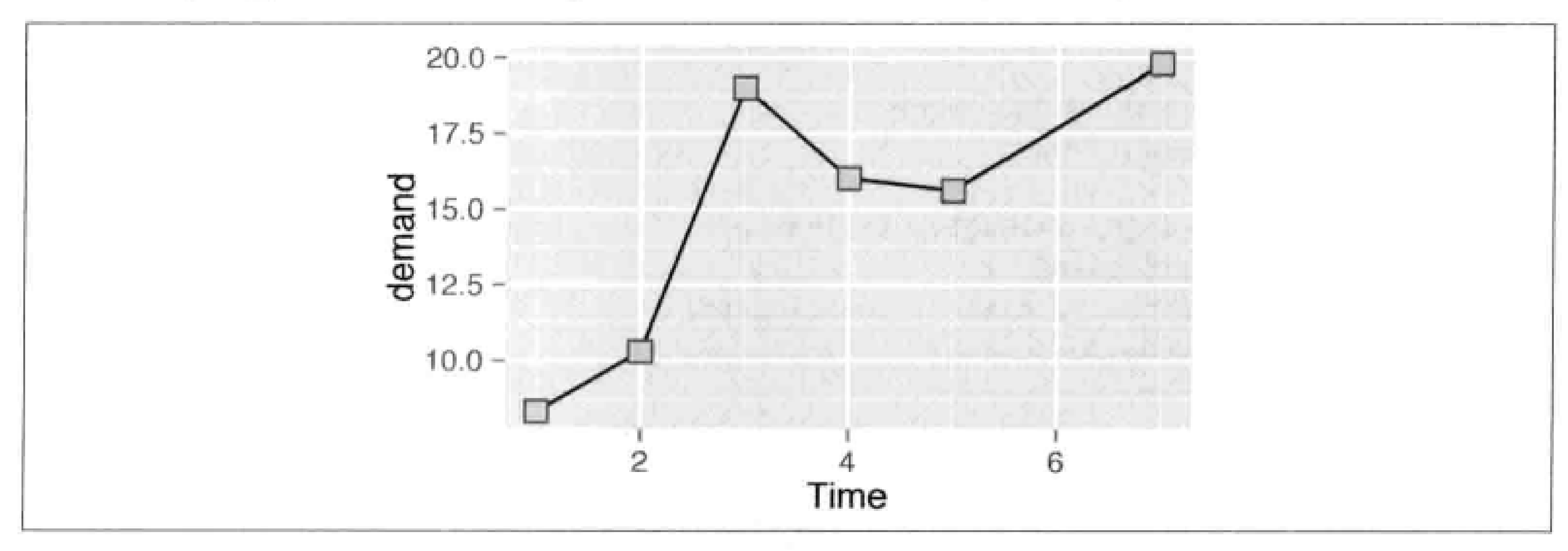


图 4-14　自定义数据标记大小、形状、颜色和填充色的折线图

讨论

数据标记默认的形状（shape）是实线圆圈，默认的大小（size）是 2，默认的颜色（colour）是黑色（black）。填充色（fill）属性只适用于某些（标号 21-25）具有独立边框线和填充颜色的点形（参见 5.3 节中的点形列表）。fill 一般取空值或者 NA。将填充色设定为白色可以得到一个空心圆，如图 4-15 所示。

```
ggplot(BOD, aes(x=Time, y=demand)) +
    geom_line() +
    geom_point(size=4, shape=21, fill="white")
```

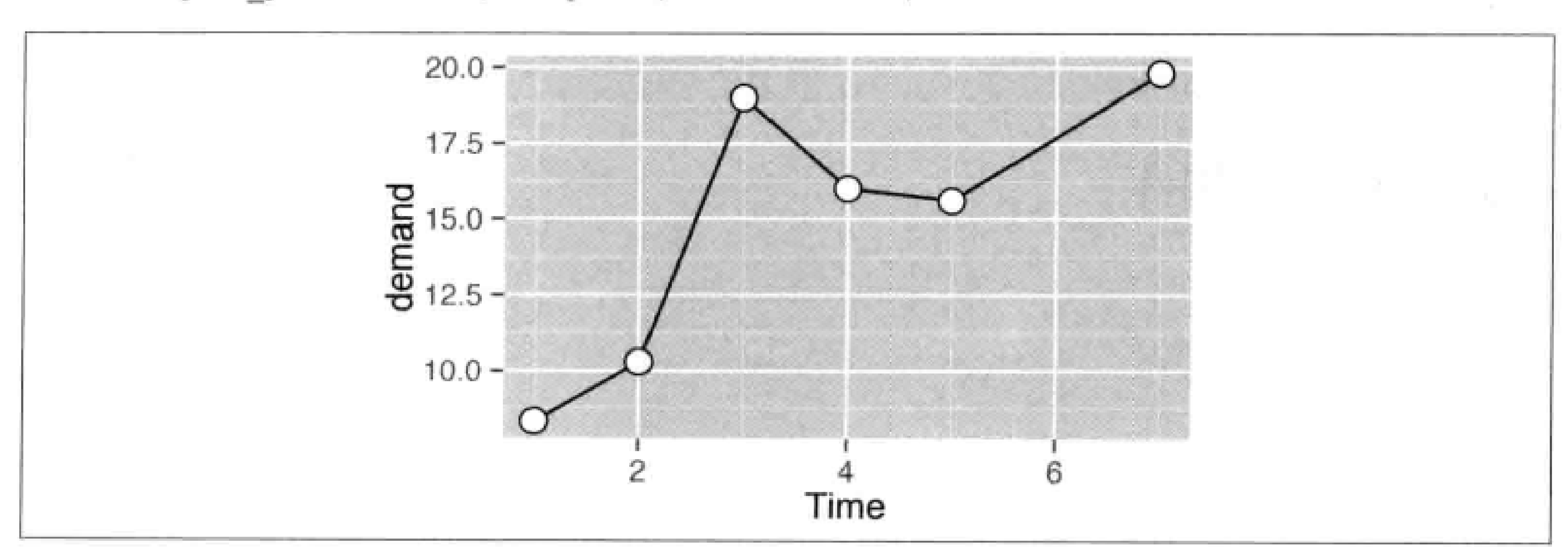


图 4-15　填充色为白色的数据标记

如果要将数据标记和折线设定为不同的颜色，我们必须在折线绘制完毕后再行设定数据标记的颜色，此时，数据标记被绘制在更上面的图层，从而，避免被折线遮盖。

从 4.3 节可知，通过在 aes() 函数内部将分组变量映射给数据标记的图形属性可以将多条折线设定为不同的颜色。数据标记的默认颜色并不吸引眼球，因而，你可能想要调用别的调色板，scale_colour_brewer() 函数和 scale_colour_manual() 函数可以完成上述操作。在 aes() 函数外部设定 shape 和 size 可以将数据标记设定为统一的形状和颜色，如图 4-16 所示。

```
# 载入 plyr 包，以使用 ddply() 函数创建例子所需数据集
library(plyr)
# 对 ToothGrowth 数据集进行汇总
tg <- ddply(ToothGrowth, c("supp", "dose"), summarise, length=mean(len))

# 保存错开（dodge）设置，接下来会多次用到
pd <- position_dodge(0.2)

ggplot(tg, aes(x=dose, y=length, fill=supp)) +
    geom_line(position=pd) +
    geom_point(shape=21, size=3, position=pd) +
    scale_fill_manual(values=c("black","white"))
```

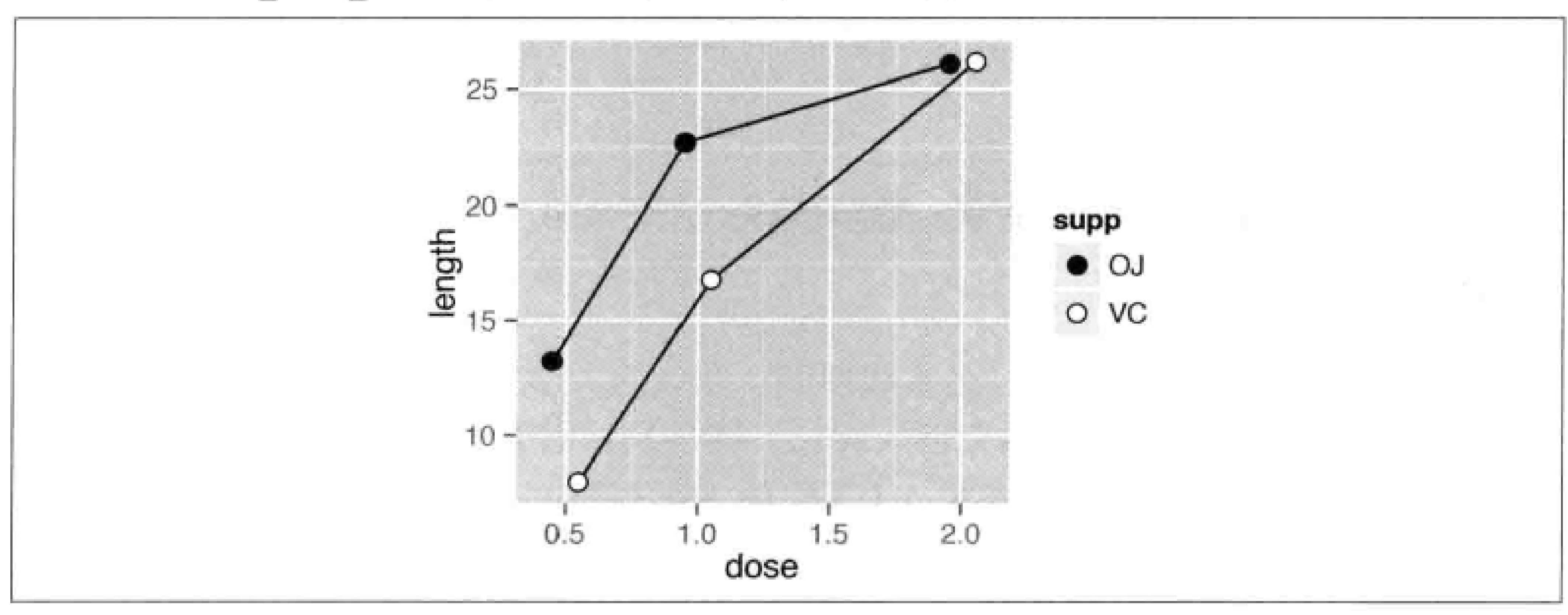


图 4-16　将填充色手动设定为黑白两色，并轻微调整数据标记的位置

另见

更多关于使用点形的内容可以参见 5.3 节，更多关于使用颜色的内容可参见本书第 12 章。

4.6　绘制面积图

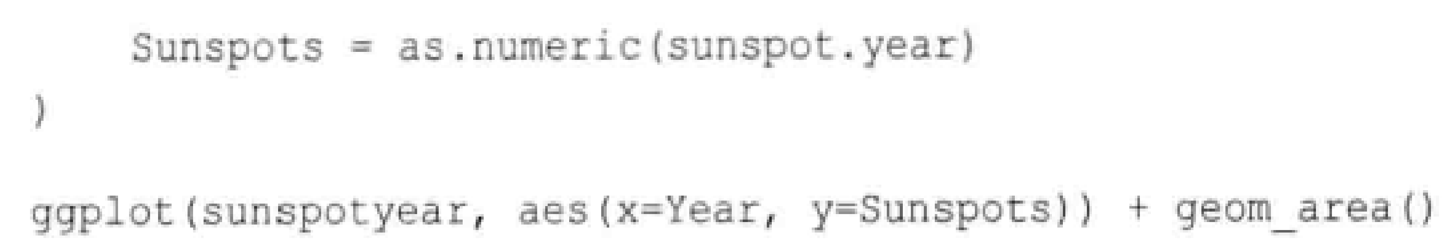
问题

如何绘制面积图？

方法

运行 geom_area() 函数即可绘制面积图，如图 4-17 所示。

```
# 将 sunspot.year 数据集转化为数据框，便于本例使用
sunspotyear <- data.frame(
    Year     = as.numeric(time(sunspot.year)),
```



```
    Sunspots = as.numeric(sunspot.year)
)

ggplot(sunspotyear, aes(x=Year, y=Sunspots)) + geom_area()
```

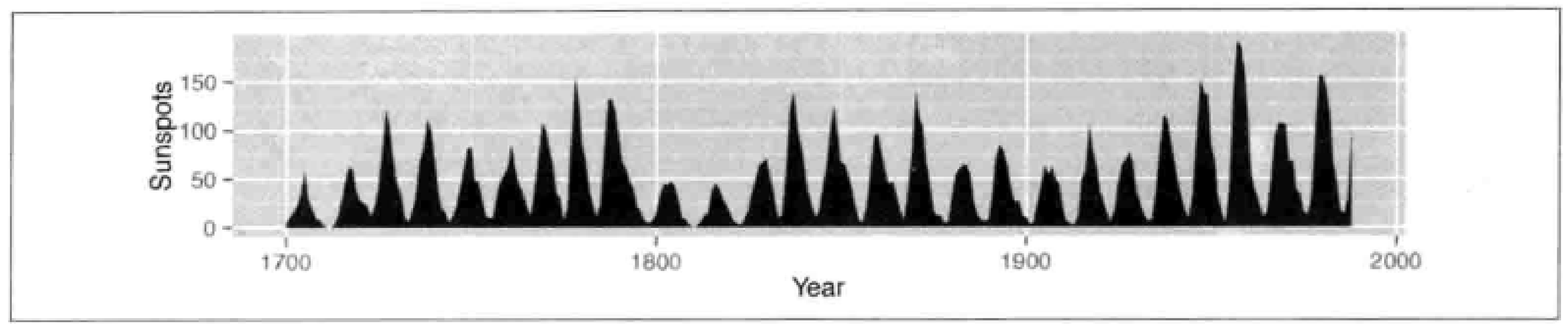


图 4-17　面积图

讨论

默认情况下，面积图的填充色为黑灰色且没有边框线，通过设定填充色（fill）可以修改面积图的填充色。接下来的例子中，我们将填充色设定为蓝色，并通过设定 alpha=0.2 将面积图的透明度设定为 80%，此时，我们可以看到面积图的网格线，如图 4-18 所示。通过设置颜色（colour）可以为面积图添加边框线：

```
ggplot(sunspotyear, aes(x=Year, y=Sunspots)) +
    geom_area(colour="black", fill="blue", alpha=.2)
```

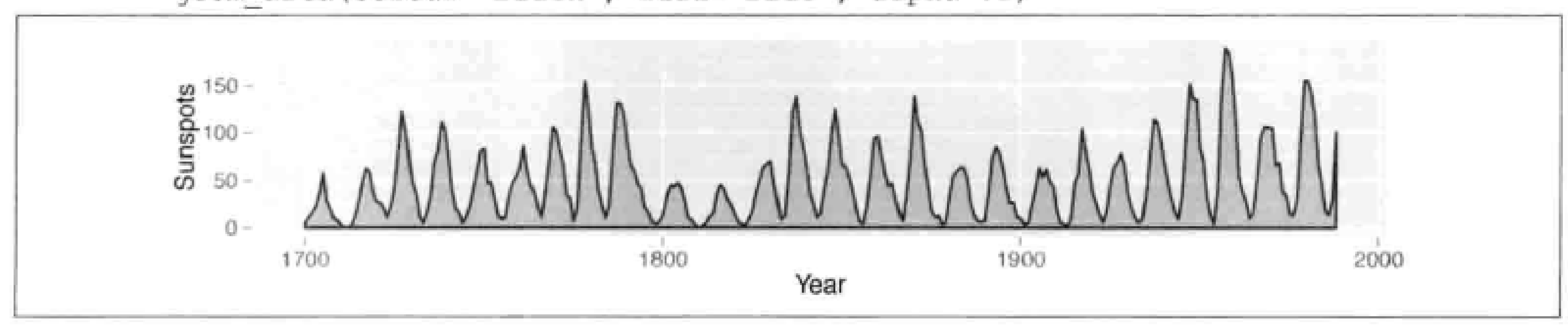


图 4-18　带半透明阴影区域和边框线的面积图

给整个面积图添加边框线之后的效果可能并不十分令人满意，因为此时系统会在面积图的起点和终点位置分别绘制一套垂直线，且在底部绘制了一条横线。为了修正上述情况，可以先绘制不带边框线的面积图（不设定 colour），然后，添加新图层，并用 geom_line() 函数绘制轨迹线，如图 4-19 所示。

```
ggplot(sunspotyear, aes(x=Year, y=Sunspots)) +
    geom_area(fill="blue", alpha=.2) +
    geom_line()
```

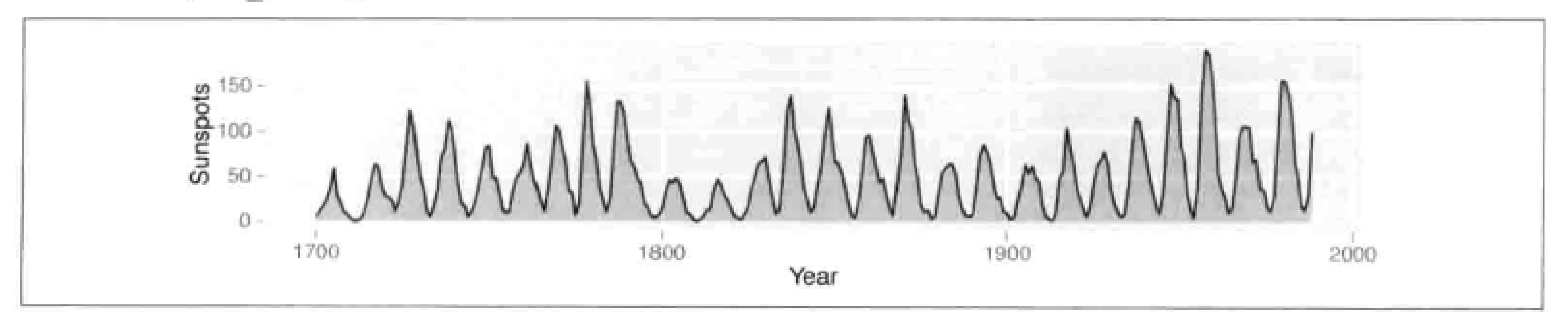


图 4-19　调用 geom_line() 函数绘制覆盖顶部的面积图

另见

更多关于使用颜色的内容可以参见本书第 12 章。

4.7 绘制堆积面积图

问题

如何绘制堆积面积图?

方法

运行 geom_area() 函数，并映射一个因子型变量给填充色（fill）即可，如图 4-20 所示。

```
library(gcookbook) # 为了使用数据

ggplot(uspopage, aes(x=Year, y=Thousands, fill=AgeGroup)) + geom_area()
```

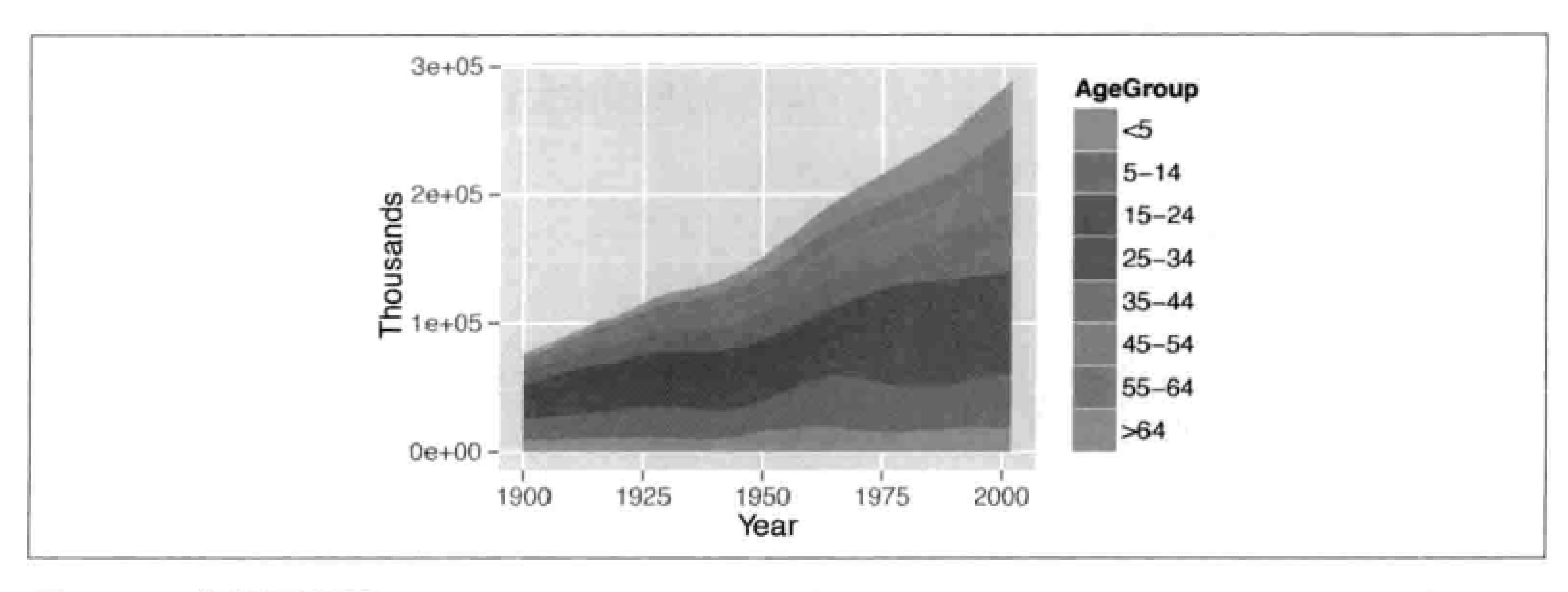


图 4-20 堆积面积图

讨论

堆积面积图对应的基础数据通常为宽格式（wide format），但 ggplot2 要求数据必须是长格式（long format），在两种格式之间进行转换的内容可参见 15.19 节。

下面以 uspopage 数据集为例：

```
uspopage

 Year AgeGroup Thousands
 1900       <5      9181
 1900     5-14     16966
 1900    15-24     14951
 1900    25-34     12161
 1900    35-44      9273
 1900    45-54      6437
 1900    55-64      4026
```

```
1900        >64       3099
1901         <5       9336
1901       5-14      17158
...
```

默认情况下图例的堆积顺序与面积图的堆积顺序是相反的。通过设定标度中的切分（breaks）参数可以翻转堆积顺序。图 4-21 中的堆积面积图对图例的堆积顺序进行了反转，将调色板设定为蓝色渐变色，并在各个区域之间添加细线（size=.2）。同时我们将填充区域设定为半透明（alpha=.4），这样可以透过填充区域看见网格线。

```
ggplot(uspopage, aes(x=Year, y=Thousands, fill=AgeGroup)) +
    geom_area(colour="black", size=.2, alpha=.4) +
    scale_fill_brewer(palette="Blues", breaks=rev(levels(uspopage$AgeGroup)))
```

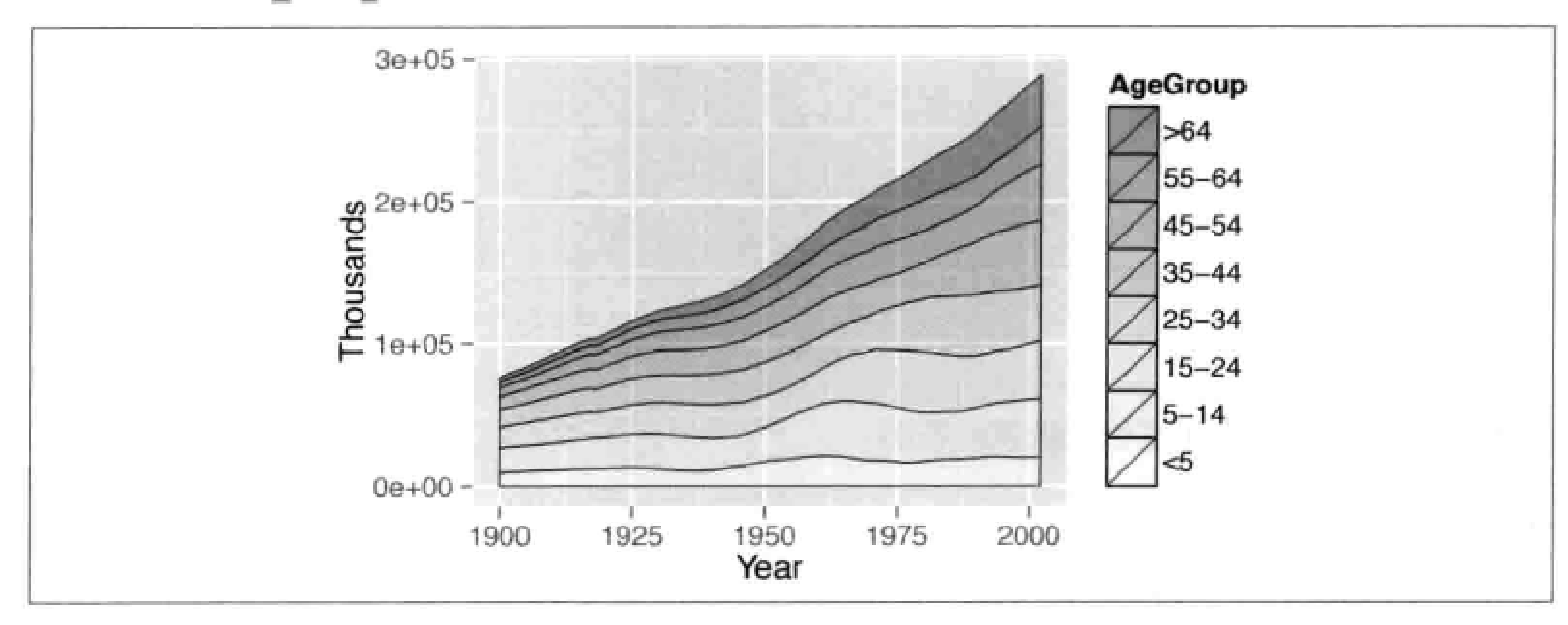


图 4-21　反转图例堆积顺序、带面积分割线并调用其他调色板的堆积面积图

在 aes() 函数内部设定 order=desc(AgeGroup) 可以对堆积面积图的堆积顺序进行反转，如图 4-22 所示。

```
library(plyr) # 为了使用 desc() 函数
ggplot(uspopage, aes(x=Year, y=Thousands, fill=AgeGroup, order=desc(AgeGroup))) +
    geom_area(colour="black", size=.2, alpha=.4) +
   scale_fill_brewer(palette="Blues")
```

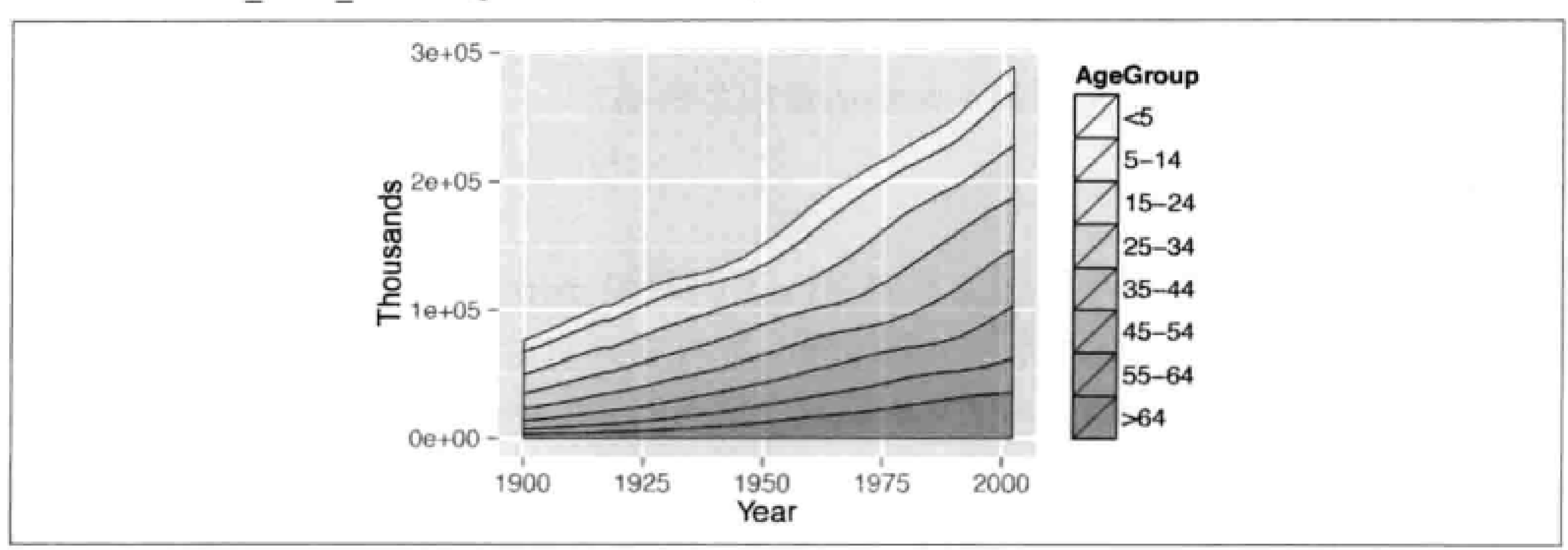


图 4-22　反转堆积顺序的堆积面积图

因为堆积面积图中的各个部分是由多边形构成的，因此其具有左、右边框线。这样的绘图效果差强人意且可能产生误导效果。为了对此进行修正（见图4-23），我们可以先绘制一个不带边框线的堆积面积图（将 colour 设定为默认的 NA 值），然后，在其顶部添加 geom_line()：

```
ggplot(uspopage, aes(x=Year, y=Thousands, fill=AgeGroup, order=desc(AgeGroup))) +
    geom_area(colour=NA, alpha=.4) +
    scale_fill_brewer(palette="Blues") +
    geom_line(position="stack", size=.2)
```

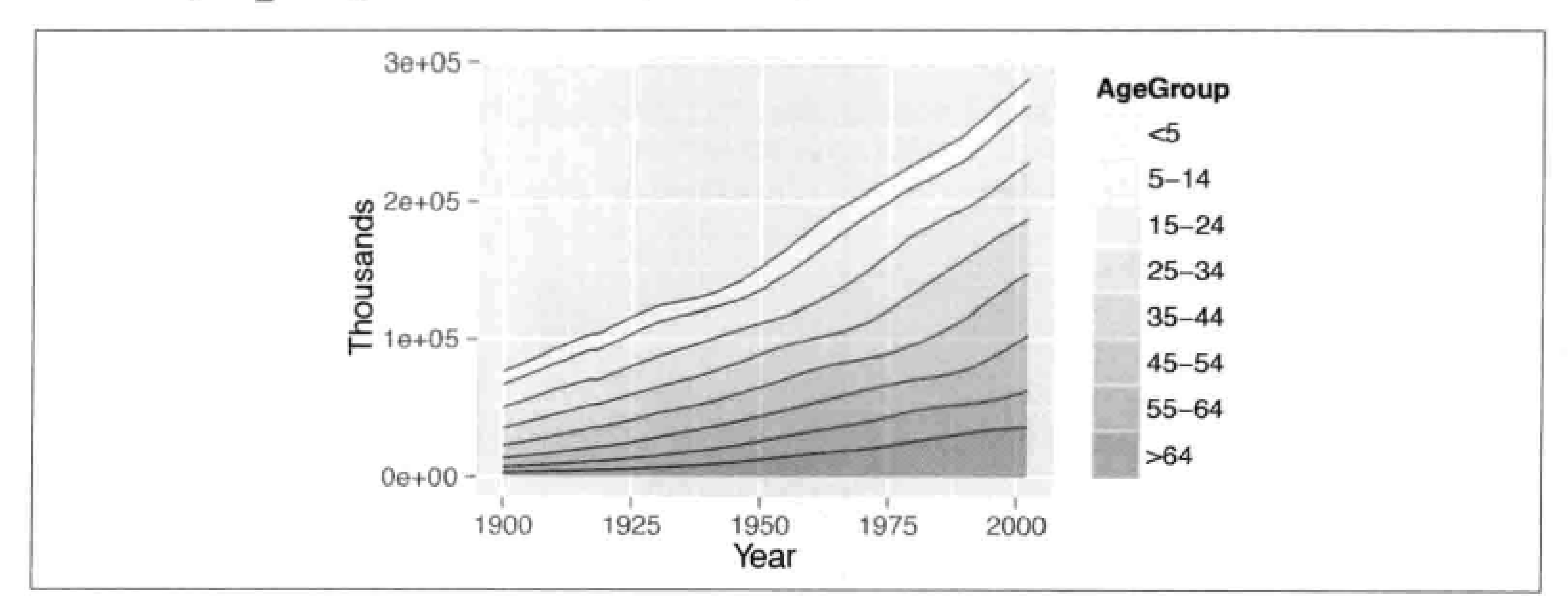


图 4-23　没有左、右边框线的堆积面积图

另见

更多关于宽格式与长格式相互转换的内容，可参见 15.3 节。

更多关于重排因子水平顺序的内容，可参见 15.8 节。

更多关于选择图形颜色的内容，可参见本书第 12 章。

4.8　绘制百分比堆积面积图

问题

如何绘制一个所有条形高度为同一常数的堆积条形图？

方法

首先，计算各组对应的百分比。本例中，我们调用 ddply() 函数按变量 Year 对 uspopage 进行分组，然后计算一个新的列，命名为 Percent。该列每一行的值等于对应的 Thousands 值除以变量 Year 对应的各个组内的 Thousands 之和再乘以 100%。

```
library(gcookbook)  # 为了使用数据
library(plyr)       # 为了使用 ddply() 函数

# 将 Thousands 转化为 Percent
```

```
uspopage_prop <- ddply(uspopage, "Year", transform,
                       Percent = Thousands / sum(Thousands) * 100)
```

计算得出百分比之后，剩余的绘图步骤与绘制普通堆积面积图的步骤一样，如图 4-24 所示。

```
ggplot(uspopage_prop, aes(x=Year, y=Percent, fill=AgeGroup)) +
    geom_area(colour="black", size=.2, alpha=.4) +
    scale_fill_brewer(palette="Blues", breaks=rev(levels(uspopage$AgeGroup)))
```

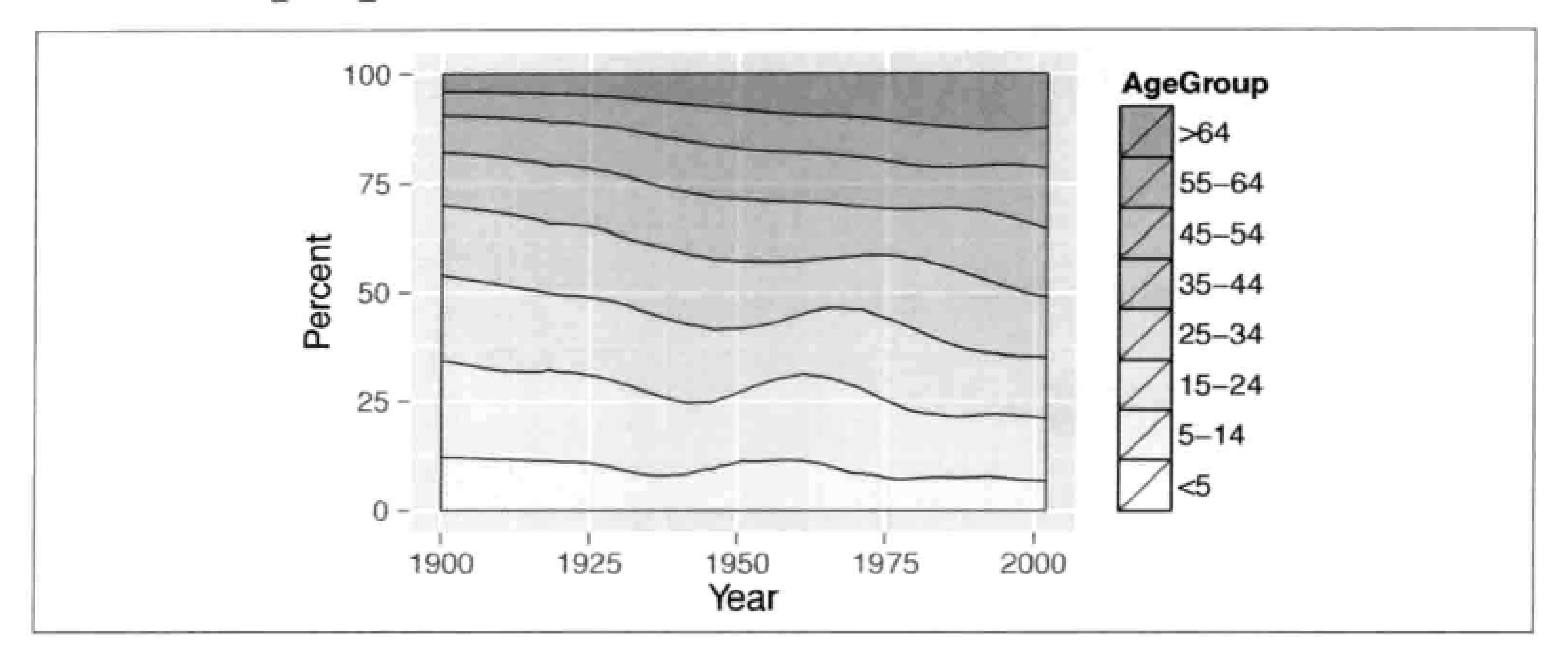


图 4-24　百分比堆积面积图

讨论

让我们更深入查看上面的数据，并探究一下数据的计算过程：

```
uspopage

 Year AgeGroup Thousands
 1900       <5      9181
 1900     5-14     16966
 1900    15-24     14951
 1900    25-34     12161
 1900    35-44      9273
 1900    45-54      6437
 1900    55-64      4026
 1900      >64      3099
 1901       <5      9336
 1901     5-14     17158
 ...
```

调用 ddply() 函数，按照变量 Year 将数据集拆分为多个独立的数据框，对所有数据框执行 transform() 函数并计算每个数据框对应的 Percent。最后，调用 ddply() 函数将所有数据框重组在一起：

```
uspopage_prop <- ddply(uspopage, "Year", transform,
                       Percent = Thousands / sum(Thousands) * 100)

 Year AgeGroup Thousands    Percent
```

```
1900        <5    9181 12.065340
1900      5-14   16966 22.296107
1900     15-24   14951 19.648067
1900     25-34   12161 15.981549
1900     35-44    9273 12.186243
1900     45-54    6437  8.459274
1900     55-64    4026  5.290825
1900       >64    3099  4.072594
1901        <5    9336 12.033409
1901      5-14   17158 22.115385
...
```

另见

更多关于分组计算数据的内容可参见 15.17 节。

4.9 添加置信域

问题

如何为折线图添加置信域？

方法

运行 geom_ribbon()，然后分别映射一个变量给 ymin 和 ymax。

climate 数据集中的 Anomaly10y 变量表示了各年温度相对于 1950—1980 平均水平变异（以摄氏度衡量）的 10 年移动平均。变量 Unc10y 表示其 95% 置信水平下的置信区间。我们令 ymax 和 ymin 分别设定为 Anomaly10y 加减 Unc10y（见图 4-25）：

```
library(gcookbook) # 为了使用数据

# 抓取 climate 数据集的一个子集
clim <- subset(climate, Source == "Berkeley",
               select=c("Year", "Anomaly10y", "Unc10y"))

clim

 Year Anomaly10y Unc10y
 1800     -0.435  0.505
 1801     -0.453  0.493
 1802     -0.460  0.486
 ...
 2003      0.869  0.028
 2004      0.884  0.029
# 将置信域绘制为阴影
ggplot(clim, aes(x=Year, y=Anomaly10y)) +
    geom_ribbon(aes(ymin=Anomaly10y-Unc10y, ymax=Anomaly10y+Unc10y),alpha=0.2) +
    geom_line()
```

阴影部分的颜色实际上是黑灰色，但看起来几乎是透明的。这是因为我们通过设定

alpha=0.2 将阴影部分的透明度设定为 80%。

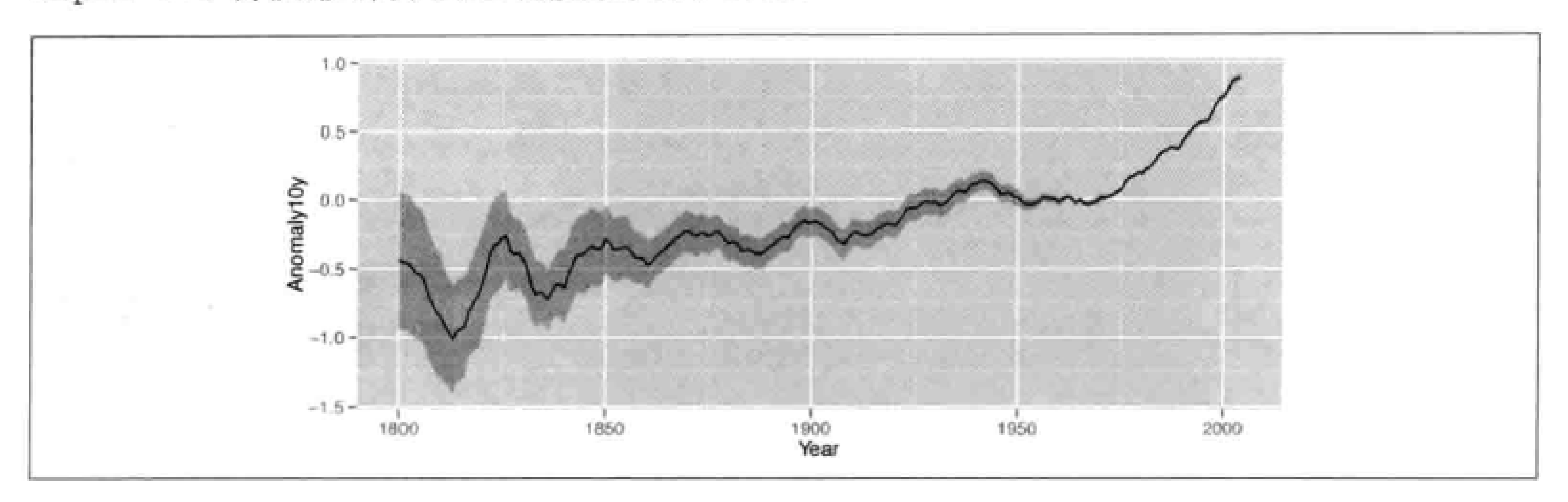


图 4-25　以阴影区域表示置信域的折线图

讨论

注意，上面的绘图命令中 geom_ribbon() 函数的调用顺序在 geom_line() 函数之前，因而，折线被绘制在阴影区域上面的图层上。如果颠倒调用顺序的话，阴影区域的颜色有可能使折线模糊不清。在本例中，似乎这不成问题，这是因为本例中的阴影区域几乎是全透明的，但当阴影区域部分不透明时这个问题就很严重。

除了使用阴影区域，我们还可以使用虚线来表示置信域的上下边界（见图 4-26）：

```
# 使用虚线表示置信域的上下边界
ggplot(clim, aes(x=Year, y=Anomaly10y)) +
    geom_line(aes(y=Anomaly10y-Unc10y), colour="grey50", linetype="dotted") +
    geom_line(aes(y=Anomaly10y+Unc10y), colour="grey50", linetype="dotted") +
    geom_line()
```

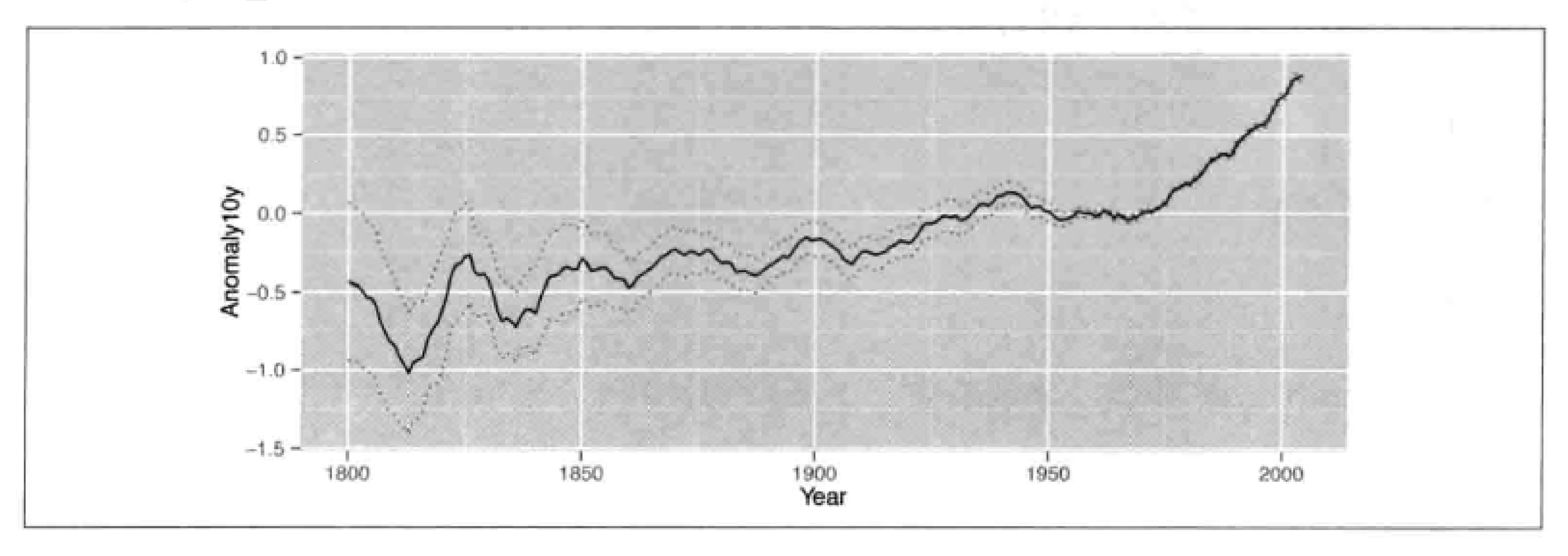


图 4-26　以虚线表示置信域的折线图

另见

除了表示置信域之外，阴影区域还可以用来表示其他内容，比如两个变量之间的差值等。

在 4.7 节中的面积图中，阴影区域的 y 轴范围是 0 到 y，而图 4-26 中 y 轴的范围是 ymin 到 ymax。

第5章

散点图

散点图通常用来刻画两个连续型变量之间的关系。绘制散点图时，数据集中的每一个观测值都由散点图中的一个点来表示。通常，人们还会向散点图中添加一些直线，以用来表示基于某些统计模型的预测值。当散点图中的数据趋势难以用肉眼识别时，这些直线对我们理解数据的特征很有帮助。上述这些操作在 R 和 ggplot2 中都是很容易做到的。

当数据集很大时，散点图上的数据点会相互重叠，此时，很难在图上清楚地显示出所有的数据点。这时候，我们可以先对数据进行加工，再绘制散点图。本章也将介绍一些加工数据的操作。

5.1 绘制基本散点图

问题

如何绘制散点图？

方法

运行 geom_point() 函数，分别映射一个变量到 x 和 y。

heightweight 是个多列数据集，接下来的例子我们只用到其中两列（见图 5-1）。

```
library(gcookbook) # 为了使用数据

# 列出我们用到的列
heightweight[, c("ageYear", "heightIn")]
```

```
 ageYear heightIn
   11.92     56.3
   12.92     62.3
   12.75     63.3
```

```
...
   13.92      62.0
   12.58      59.3
```

```
ggplot(heightweight, aes(x=ageYear, y=heightIn)) + geom_point()
```

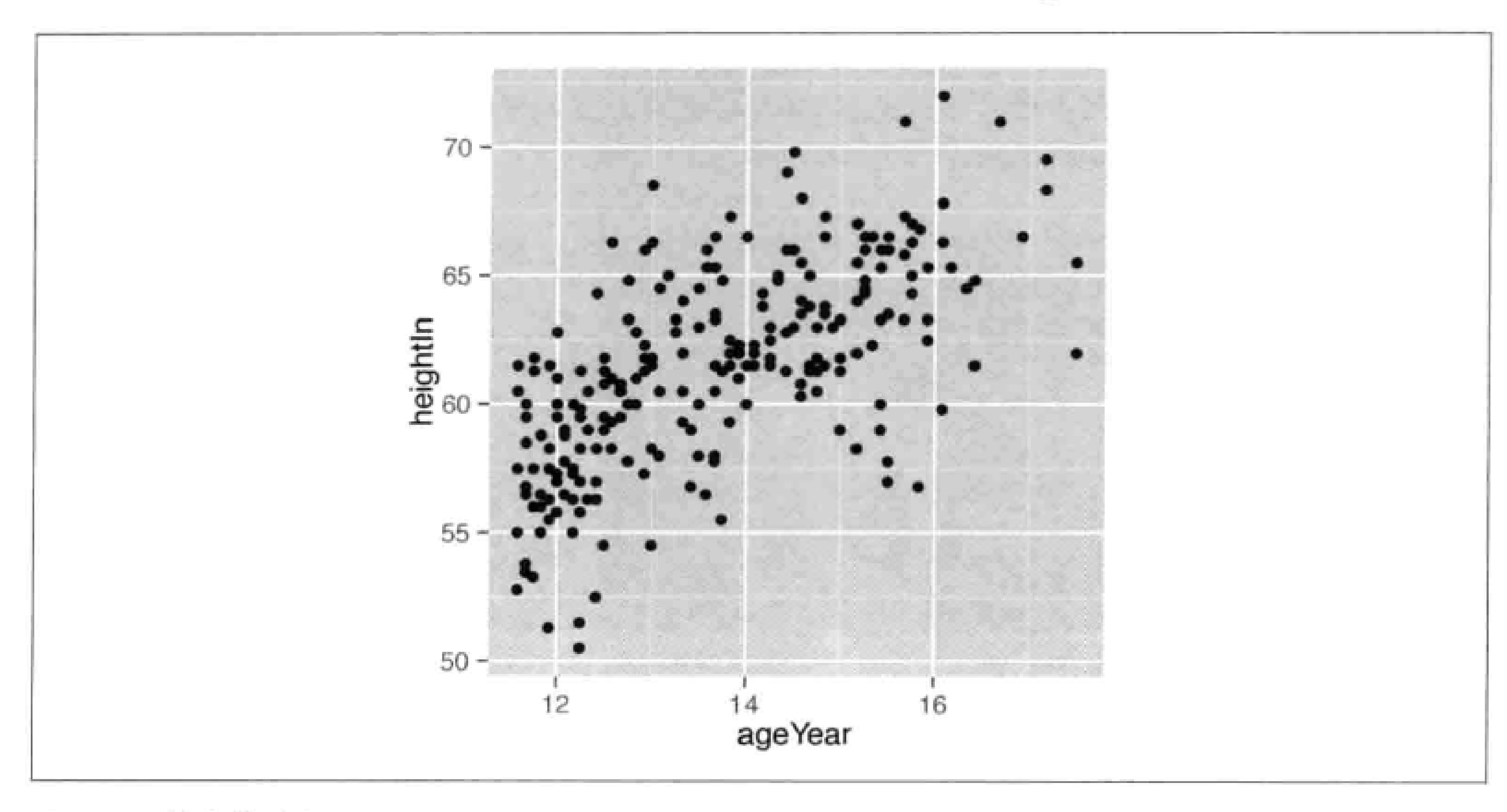


图 5-1 基本散点图

讨论

通过设定点形（shape）参数可以在散点图中绘制默认值以外的点形。比如，我们常用空心圈（点形 21）代替实心圆（点形 16），如图 5-2 左图所示：

```
ggplot(heightweight, aes(x=ageYear, y=heightIn)) + geom_point(shape=21)
```

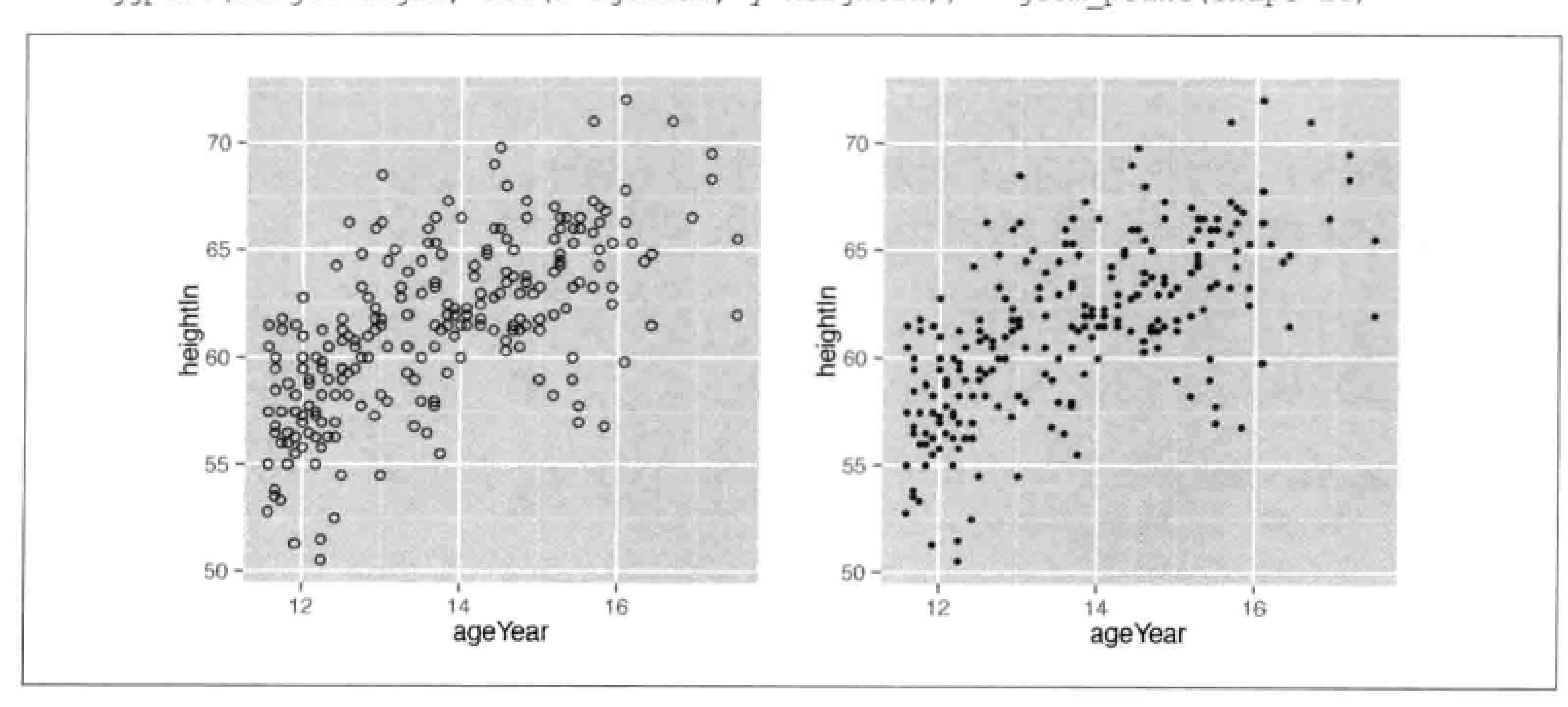


图 5-2 左图：空心圆散点图（即点形 21） 右图：数据点更小的的散点图

大小（size）参数可以控制图中点的大小。系统默认的大小（size）值等于 2，下面我们将其设定为 size=1.5，以得到更小的数据点（见图 5-2 右图）：

```
ggplot(heightweight, aes(x=ageYear, y=heightIn)) + geom_point(size=1.5)
```



在某些系统平台上，点形 16 显示在屏幕上或者输出到 PNG 等位图文件时会显示锯齿状边沿。此时，我们可用点形 19 代替点形 16 对其进行修正。虽然两者同为实心圆，但大多数时候，点形 19 的输出结果显示得较为平滑（见图 5-3）。更多关于抗锯齿（anti-aliased）的内容可参见 14.5 节。

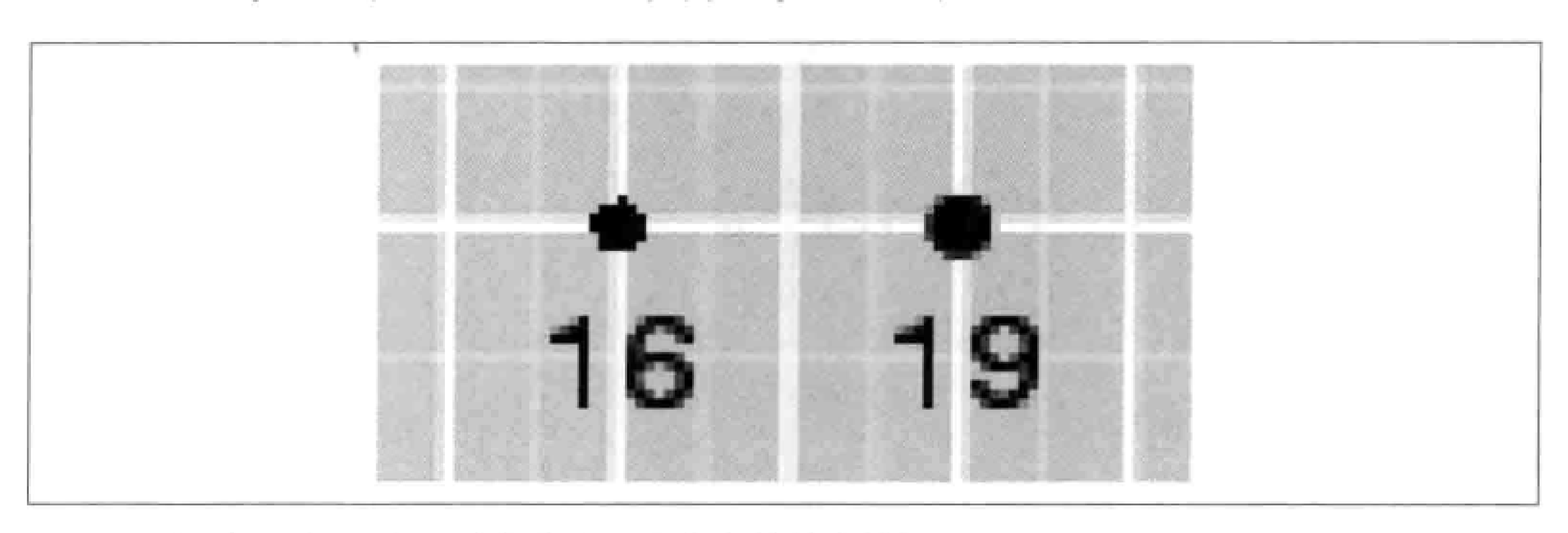


图 5-3　点形 16 和点形 19 在某些位图设备上的输出结果

5.2　使用点形和颜色属性，并基于某变量对数据进行分组

问题

如何基于某个变量对数据进行分组，并用形状和颜色属性来表示？

方法

将分组变量映射给点形（shape）和颜色（colour）属性。heightweight 是个多列数据集，接下来的例子中，我们只用其中三列：

```
library(gcookbook) # 为了使用数据
# 列出要用的三个列
heightweight[, c("sex", "ageYear", "heightIn")]

 sex ageYear heightIn
   f   11.92     56.3
   f   12.92     62.3
   f   12.75     63.3
...
   m   13.92     62.0
   m   12.58     59.3
```

通过将变量 sex 映射给 colour 或 shape，我们可以按变量 sex 对数据点进行分组（见图 5-4）：

```
ggplot(heightweight, aes(x=ageYear, y=heightIn, colour=sex)) + geom_point()

ggplot(heightweight, aes(x=ageYear, y=heightIn, shape=sex)) + geom_point()
```

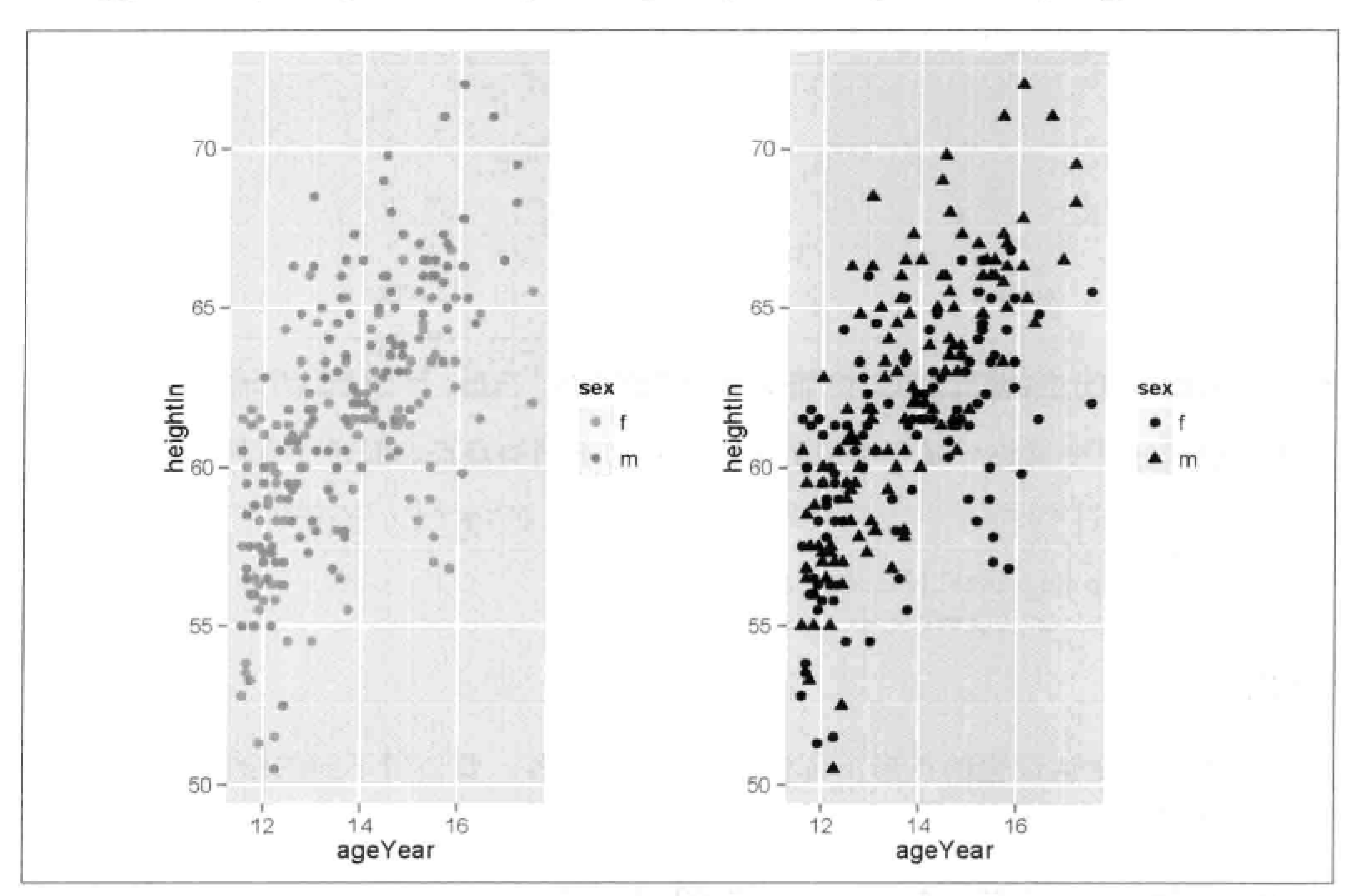


图 5-4　左图：按映射 colour 的变量对数据进行分组　右图：按映射给 shape 的变量对数据进行分组

讨论

分组变量必须是分类变量，换言之，它必须是因子型或者字符串型的向量。如果分组变量以数值型变量进行存储，则需要将它转化为因子型变量之后，才能以其作为分组变量。

可以将一个变量同时映射给 shape 和 colour 属性。当有多个分组变量时，可以将它们分别映射给这两个图形属性。下面，我们把 sex 变量同时映射给 shape 和 colour 属性（见图 5-5 左图）：

```
ggplot(heightweight, aes(x=ageYear, y=heightIn, shape=sex, colour=sex)) +
    geom_point()
```

散点图默认的点形和颜色可能不是很吸引人，通过调用 scale_shape_manual() 函数可以使用其他点形；调用 scale_colour_brewer() 或者 scale_colour_manual() 函数可以使用其他调色板。

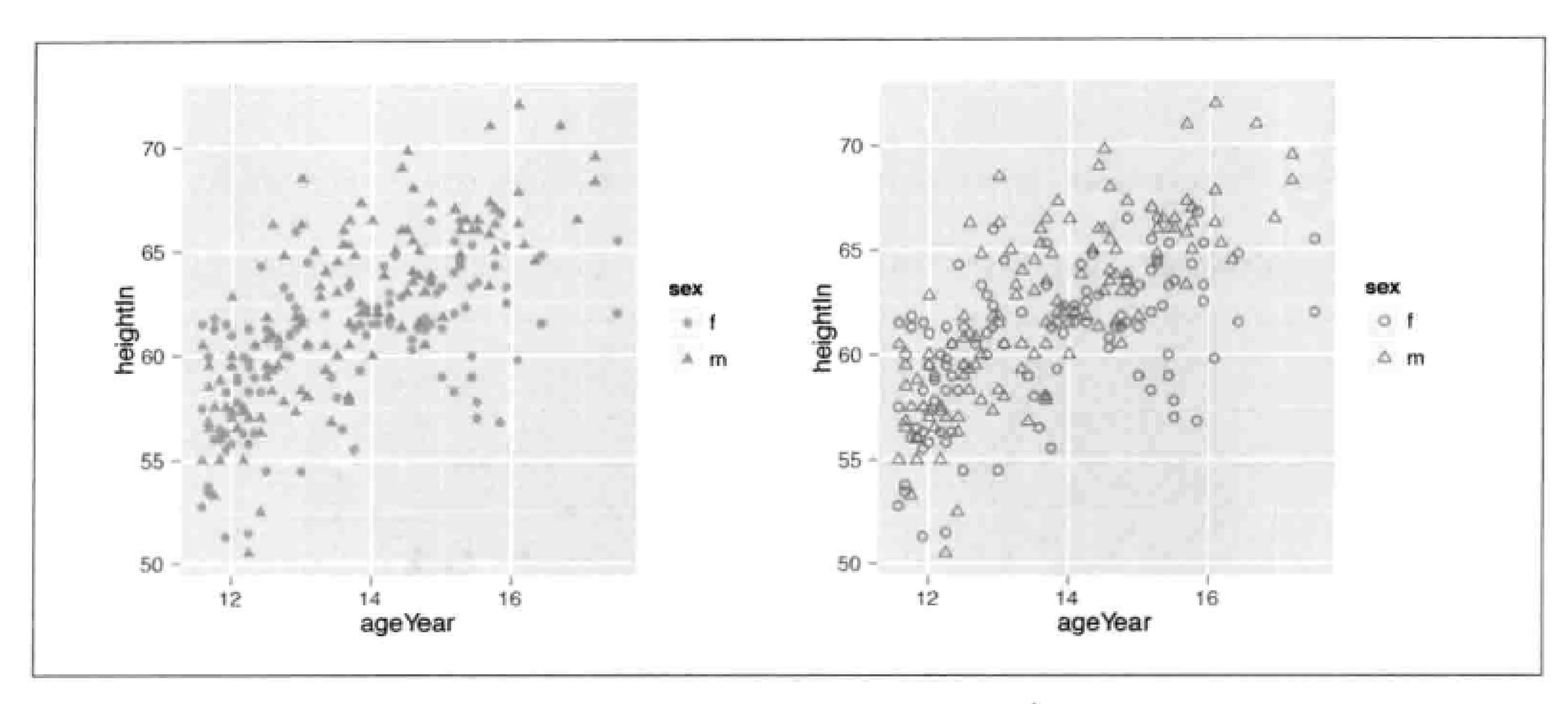


图 5-5　左图：将分组变量同时映射给点形和颜色的散点图　右图：手动设置点形和颜色的散点图

系统会根据分组变量将分属各组的数据点设置为不同的点形和颜色（见图 5-5 右图）：

```
ggplot(heightweight, aes(x=ageYear, y=heightIn, shape=sex, colour=sex)) +
    geom_point() +
    scale_shape_manual(values=c(1,2)) +
    scale_colour_brewer(palette="Set1")
```

另见

要使用不同于默认设置的点形，可参见 5.3 节的内容。要使用不同的绘图颜色，可参见本书第 12 章的内容。

5.3　使用不同于默认设置的点形

问题

如何在散点图中使用不同于默认值的点形？

方法

通过指定 geom_point() 函数中的点形（shape）参数可以设定散点图中所有数据点的点形（见图 5-6）：

```
library(gcookbook) # 为了使用数据

ggplot(heightweight, aes(x=ageYear, y=heightIn)) + geom_point(shape=3)
```

如果已将分组变量映射给 shape，则可以调用 scale_shape_manual() 函数来修改点形：

```
# 使用略大且自定义点形的数据点
ggplot(heightweight, aes(x=ageYear, y=heightIn, shape=sex)) +
    geom_point(size=3) + scale_shape_manual(values=c(1, 4))
```

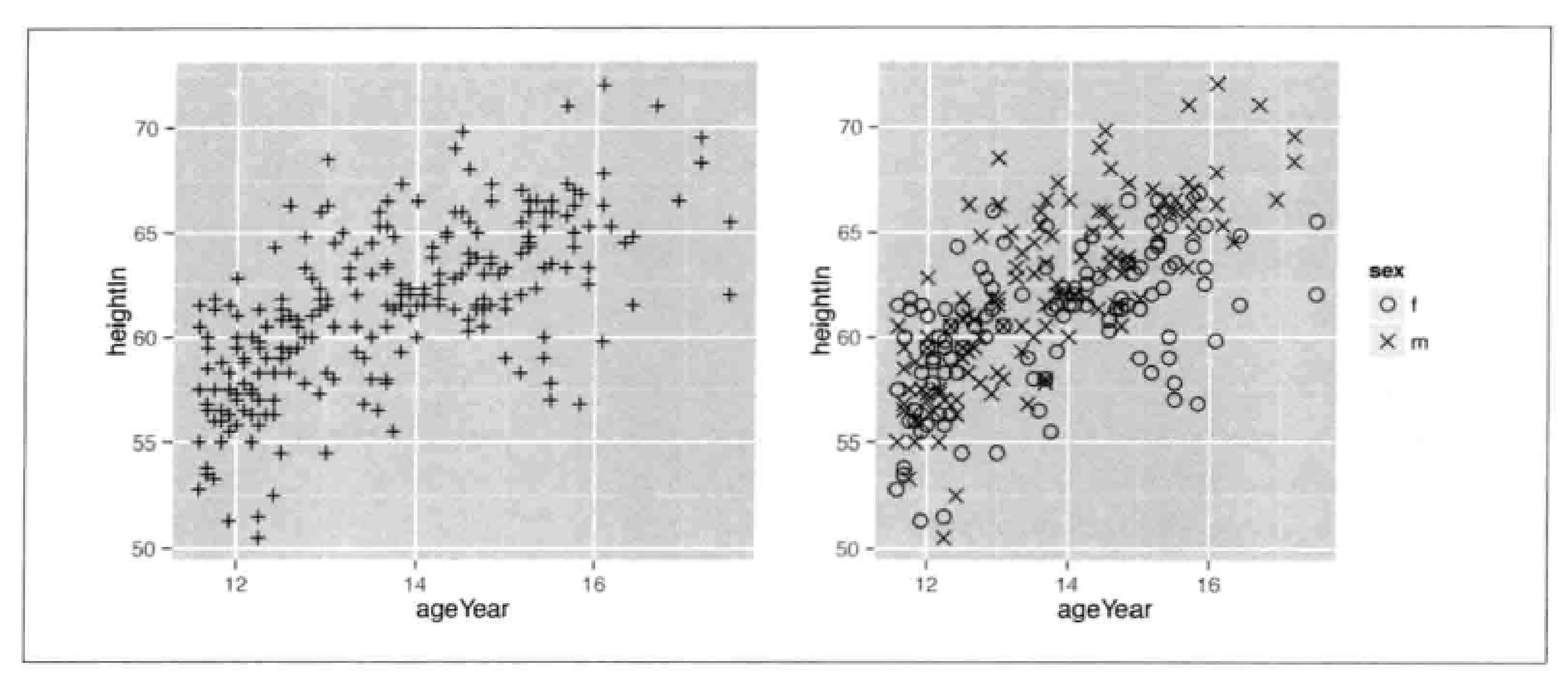


图 5-6　左图：自定义点形属性的散点图　右图：将分组变量映射给自定义点形属性的散点图

讨论

图 5-7 显示了 R 绘图中可调用的点形。其中一些点形只有边框线；一些只有实心区域；还有一些则是由可分离的边框线和具有填充色的实心区域共同组成（我们也可以用字符作点形）。

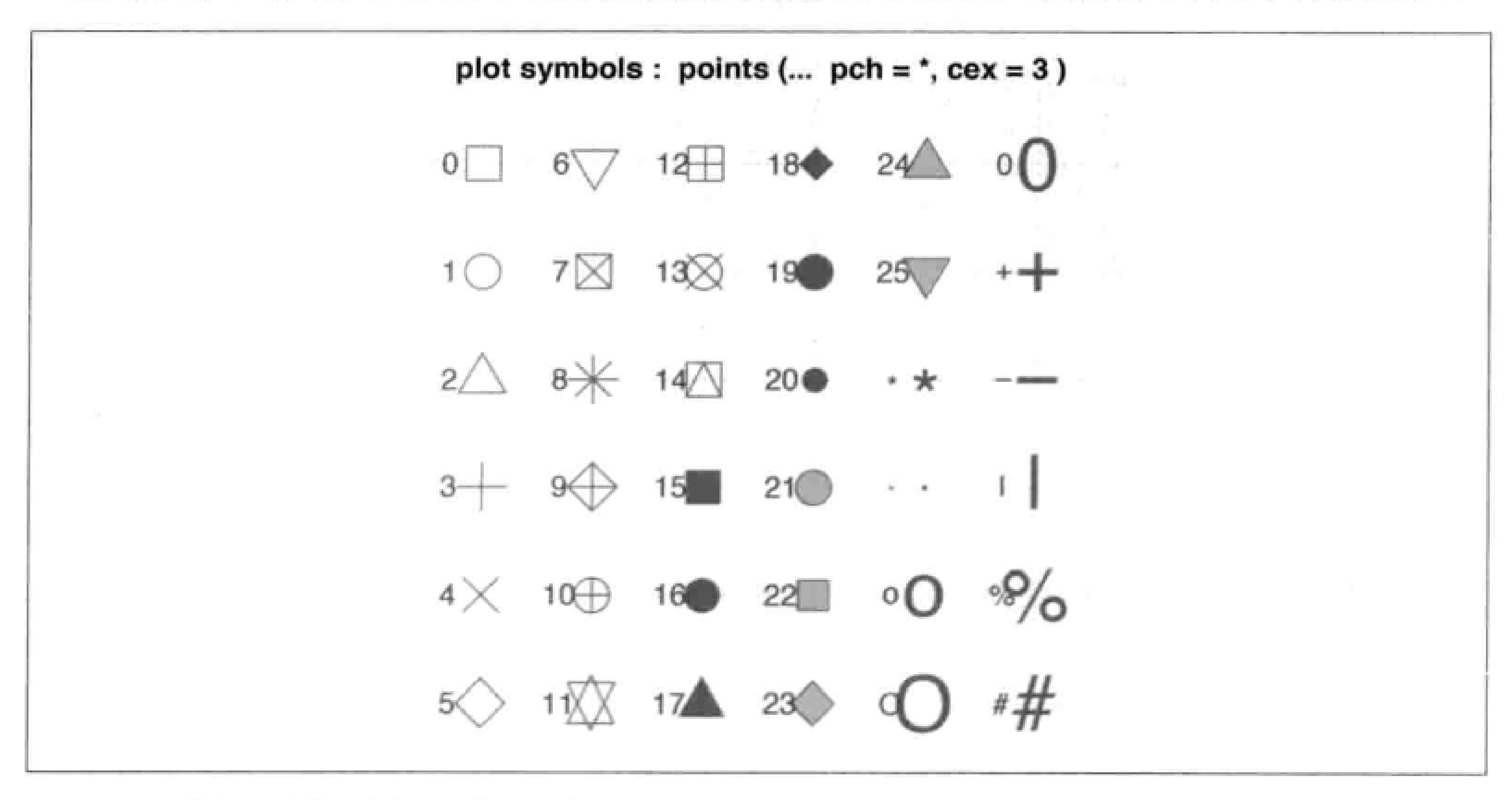


图 5-7　R 绘图系统可以调用的点形

点形 1-20 的点的颜色，包括实心区域的颜色都可由 `colour` 参数来控制。对于点形 21-25 而言，边框线和实心区域的颜色则分别由 `colour` 和 `fill` 参数来控制。

我们可以用点形和填充色（空心或实心）属性分别表示两个不同的变量。但这一过程不太直接，我们要选择一个同时具有 `colour` 和 `fill` 属性的点形及一个包括 `NA` 和其他颜色的调色板（`NA` 会生成一个空心的形状）。下面以 `heightweight` 数据集为例，同

时在数据集中增加一个用来标识儿童体重是否超过 100 磅的列（见图 5-8）：

```
# 生成一个数据副本
hw <- heightweight
# 将数据按照是否大于 100 磅分为两组
hw$weightGroup <- cut(hw$weightLb, breaks=c(-Inf, 100, Inf),
                      labels=c("< 100", ">= 100"))

# 使用具有颜色和填充色的点形及对应于空值（NA）和填充色的颜色
ggplot(hw, aes(x=ageYear, y=heightIn, shape=sex, fill=weightGroup)) +
    geom_point(size=2.5) +
    scale_shape_manual(values=c(21, 24)) +
    scale_fill_manual(values=c(NA, "black"),
      guide=guide_legend(override.aes=list(shape=21)))
```

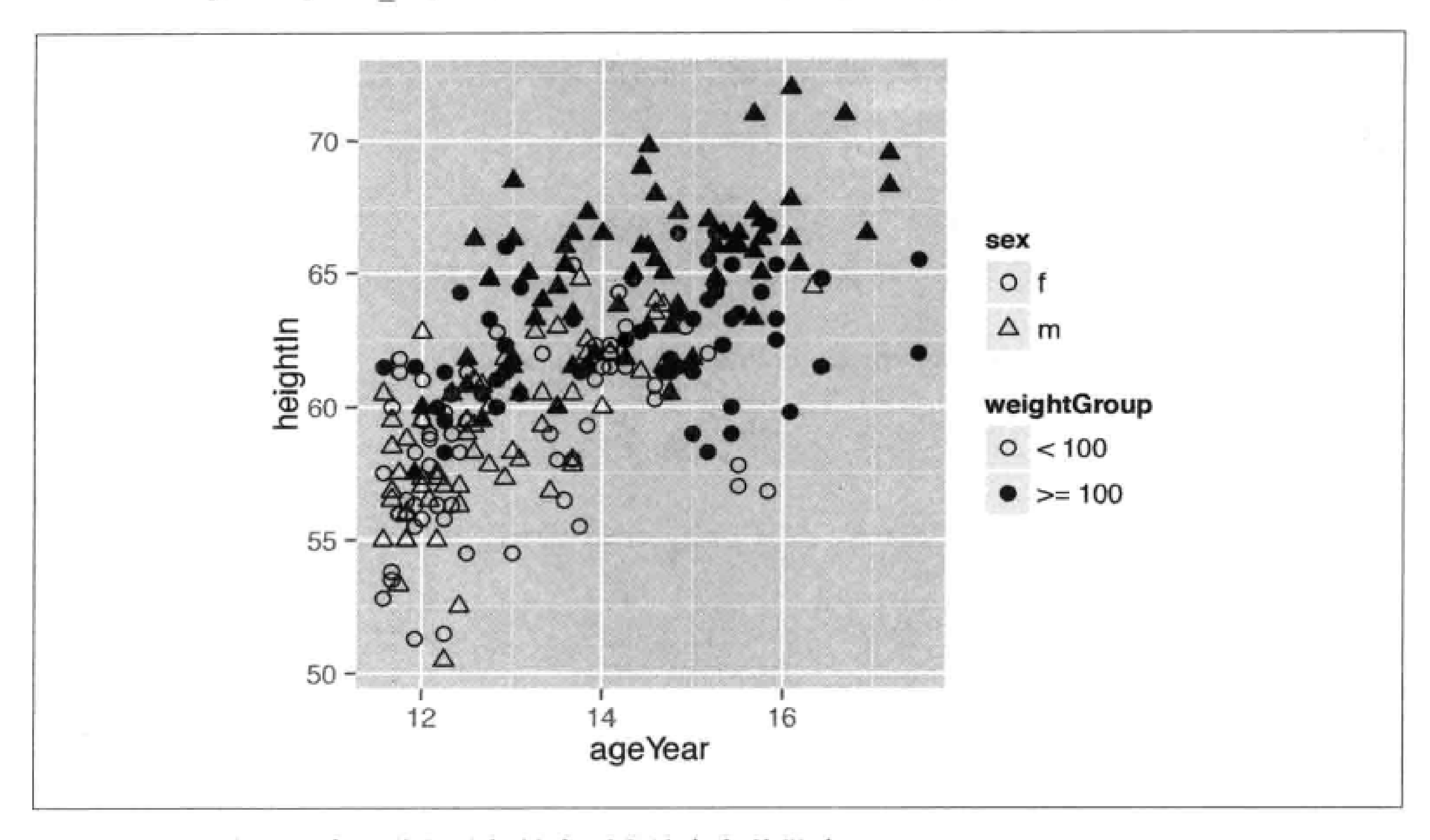


图 5-8　把两个分组变量分别映射给点形和填充色的散点图

另见

更多关于使用不同颜色的内容，可参见本书第 12 章。更多关于将连续型变量重编码为分组变量的内容，可参见 15.14 节。

5.4　将连续型变量映射到点的颜色或大小属性上

问题

如何使用散点图中的颜色和大小属性来刻画第三个连续型变量？

方法

将连续型变量映射到 size 或 colour 属性上即可。heigthweight 数据集有很多列，下面的例子中只用其中四列：

```
library(gcookbook) # 为了使用数据

# 列出要用到的四列
heightweight[, c("sex", "ageYear", "heightIn", "weightLb")]

 sex ageYear heightIn weightLb
   f   11.92     56.3     85.0
   f   12.92     62.3    105.0
   f   12.75     63.3    108.0
...
   m   13.92     62.0    107.5
   m   12.58     59.3     87.0
```

5.1 节中的散点图刻画了两个连续型变量 ageYear 和 heightIn 的关系。如果想要表示第三个连续型变量 WeightLb，必须将其映射给其他的图形属性，例如，colour 和 size，如图 5-9 所示。

```
ggplot(heightweight, aes(x=ageYear, y=heightIn, colour=weightLb)) + geom_point()

ggplot(heightweight, aes(x=ageYear, y=heightIn, size=weightLb)) + geom_point()
```

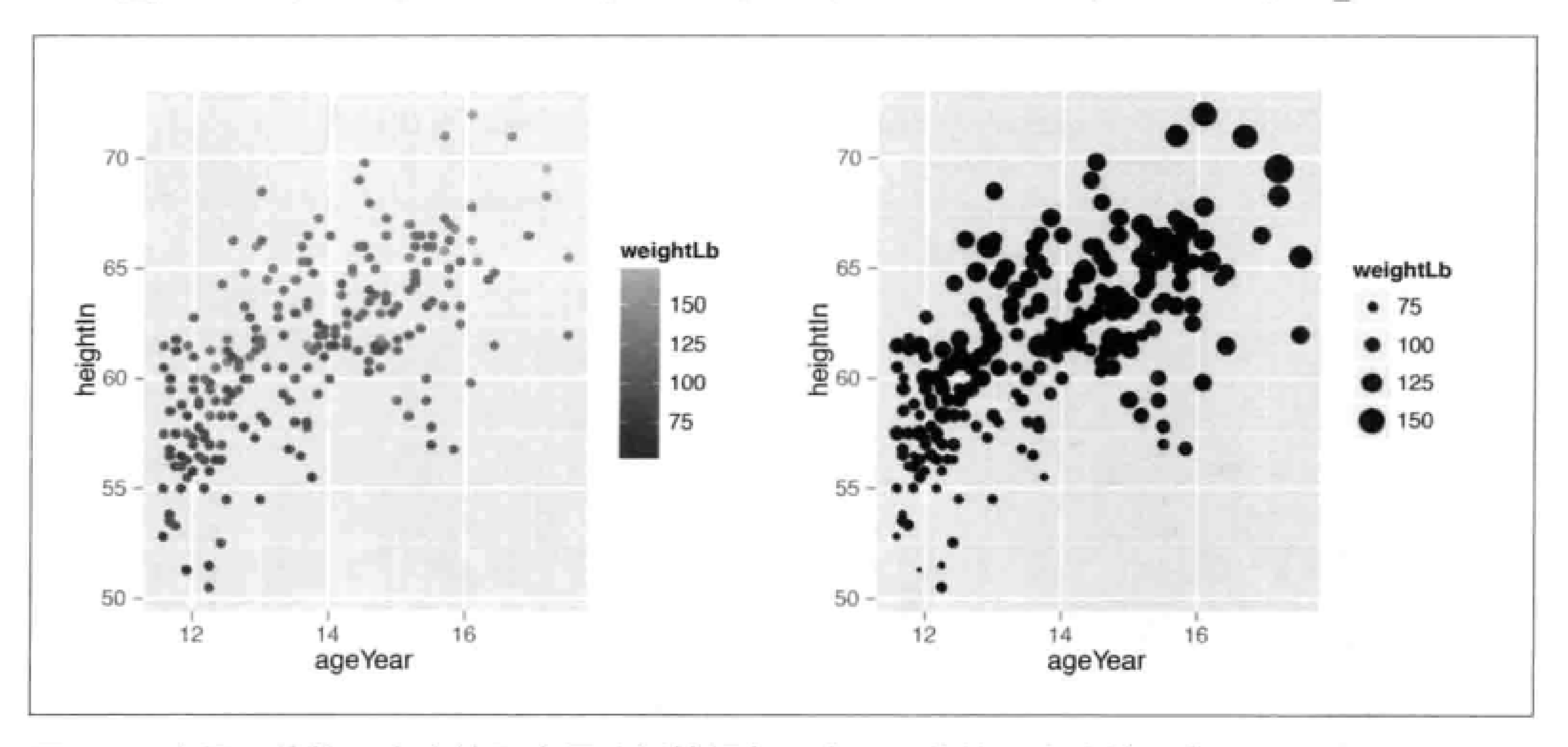


图 5-9　左图：将第三个连续型变量映射给颜色　右图：将第三个连续型变量映射给大小

讨论

基本散点图通过将两个连续型变量分别映射给 x 轴和 y 轴来刻画它们之间的关系。当变量超过两个时，我们必须将它们映射到其他图形属性上，如数据点的大小和颜色。

人类对于感知空间位置的微小变化很擅长，因此，我们可以以较高的精度解释被映射到 x 轴和 y 轴上的变量。但我们对于图形颜色和大小的变化不太敏锐，所以，我们只能以较低的精度对映射到这些属性上的变量进行解释。因此，只有当一个变量不需要高精度的解释时，它才适合被映射给图形的大小和颜色属性。

当将变量映射给大小（size）属性时，绘制的图形结果常常具有误导性。比如在图 5-9 中，最大点对应的面积是最小点所对应面积的 36 倍，然而，前者对应的变量值仅为后者的 3.5 倍。如果点的大小正比于变量值对于图形展示很重要的话，则可以修改一下数据点大小的变化范围。默认情况下，点的大小为 1 ～ 6 ms。运行 scale_size_continuous(range=c(2, 5)) 可以将其修改为 2 ～ 5 ms。然而，由于点的大小与点的直径或者面积是非线性映射，所以，这个表示值依然不精确（使点的面积与变量值成正比的相关细节，可以参见 5.12 节的内容）。

对于颜色，实际上有两个相关的图形属性可以使用，即 colour 和 fill。对于大多数点形，我们都是通过 colour 属性设定颜色。然而，点形 21-25 除了实心区域还有边框线，此时实心区域的颜色由 fill 来控制。当数据点颜色较浅时，带边框线的点形就显得非常有用，因为此时边框线可以将数据点与背景色区分开，如图 5-10 所示。本例中将色阶设定为由黑至白，同时增加数据点的大小，以便于看出填充色。

```
ggplot(heightweight, aes(x=ageYear, y=heightIn, fill=weightLb)) +
    geom_point(shape=21, size=2.5) +
    scale_fill_gradient(low="black", high="white")

# 使用 guide_legend() 函数以离散的图例代替色阶
ggplot(heightweight, aes(x=ageYear, y=heightIn, fill=weightLb)) +
    geom_point(shape=21, size=2.5) +
    scale_fill_gradient(low="black", high="white", breaks=seq(70,170,by=20),
                        guide=guide_legend())
```

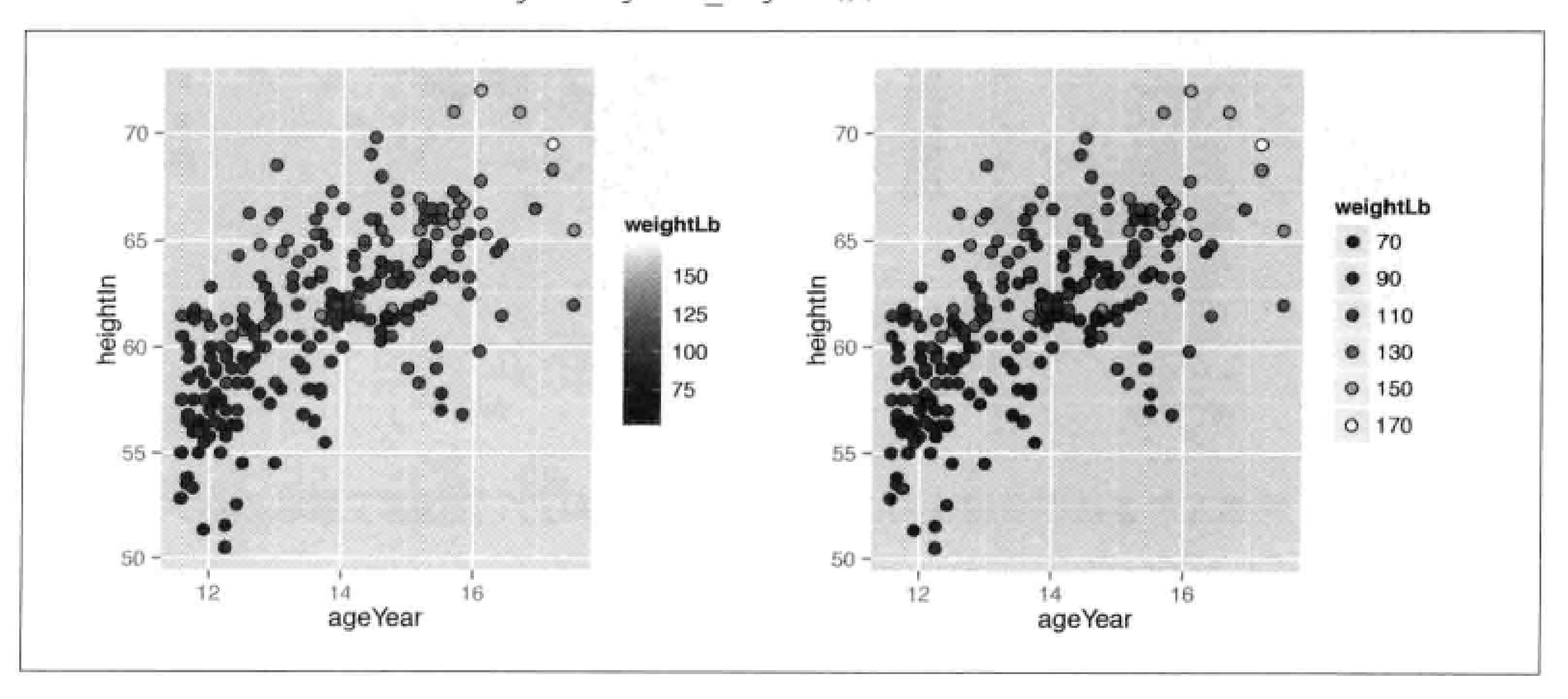


图 5-10　左图：带边框线、将连续型变量映射给 fill 的散点图　右图：使用离散型的图例代替连续型色阶

当我们把一个连续型变量映射给某个图形属性之后，这并不妨碍我们同时将分类变量映射给其他图形属性。图 5-11 中，我们将变量 weightLb 映射给点 size 属性，同时将变量 sex 映射给 colour 属性。图形中有很多重合的数据点，因此，我们设定 alpha=.5 将数据点设定为半透明。调用 scale_size_area() 函数使数据点的面积正比于变量值（参见 5.12 节），同时，修改调色板使图形更吸引眼球：

```
ggplot(heightweight, aes(x=ageYear, y=heightIn, size=weightLb, colour=sex)) +
    geom_point(alpha=.5) +
    scale_size_area() +     # 使数据点面积正比于变量值
    scale_colour_brewer(palette="Set1")
```

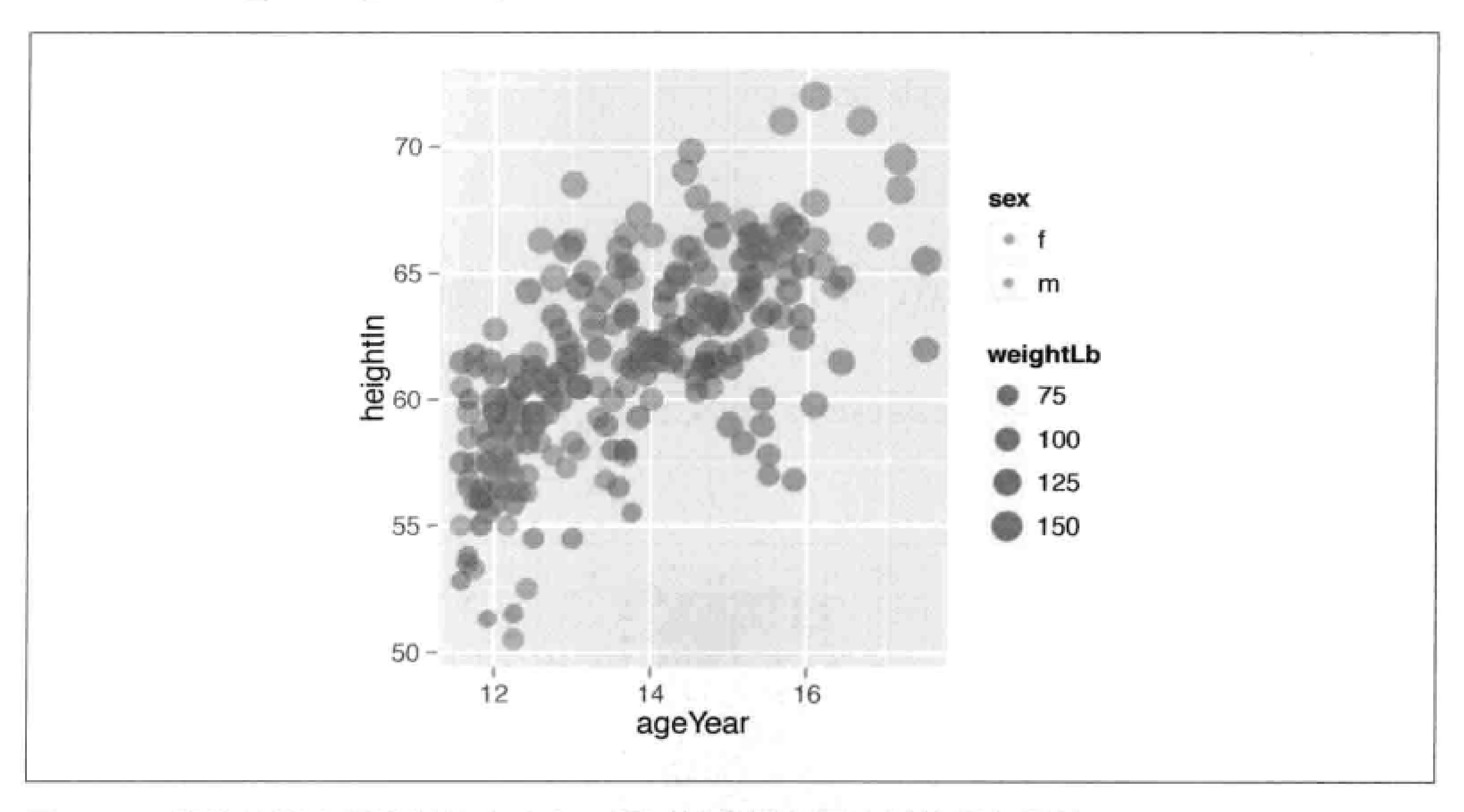


图 5-11　将连续型变量映射给点大小，同时将分类变量映射给颜色属性

将某个变量映射给 size 属性时，最好避免将其他变量映射给 shape 属性。因为不同点形的点大小很难相互比较。比如，大小为 4 的三角形看起来比大小为 3.5 的圆形更小。同时，有些形状本身就具有不同的大小：点形 16 和点形 19 都是圆形，但无论点大小设定为多少，点形 19 的圆总是比点形 16 的圆看起来更大。

另见

使用与默认设置不同的颜色，可参考 12.6 节的内容。关于创建气泡图的内容可参见 5.12 节。

5.5　处理图形重叠

问题

散点图中有大量数据点时，如何避免它们互相重叠？

方法

针对大数据集绘制散点图时，图中各个数据点会彼此遮盖，从而妨碍我们准确地评估数据的分布信息，这就是所谓的图形重叠（overplotting）。如果图形的重叠程度较低，我们可以通过使用较小的数据点或者使用不会遮盖其他数据点的点形（例如 1 号的空心圆）来避免数据重叠。5.1 节中的图 5-2 就对这两种解决方案都进行了演示。

如果图形的重叠程度较高，下面是一系列可行的解决方案：

- 使用半透明的点；
- 将数据分箱（bin），并用矩形表示（适用于量化分析）；
- 将数据分箱（bin），并用六边形表示；
- 使用箱线图。

讨论

图 5-12 中包含 54 000 个数据点，各个数据点重叠严重，我们很难看清楚图中不同区域的数据点的相对密度：

```
sp <- ggplot(diamonds, aes(x=carat, y=price))

sp + geom_point()
```

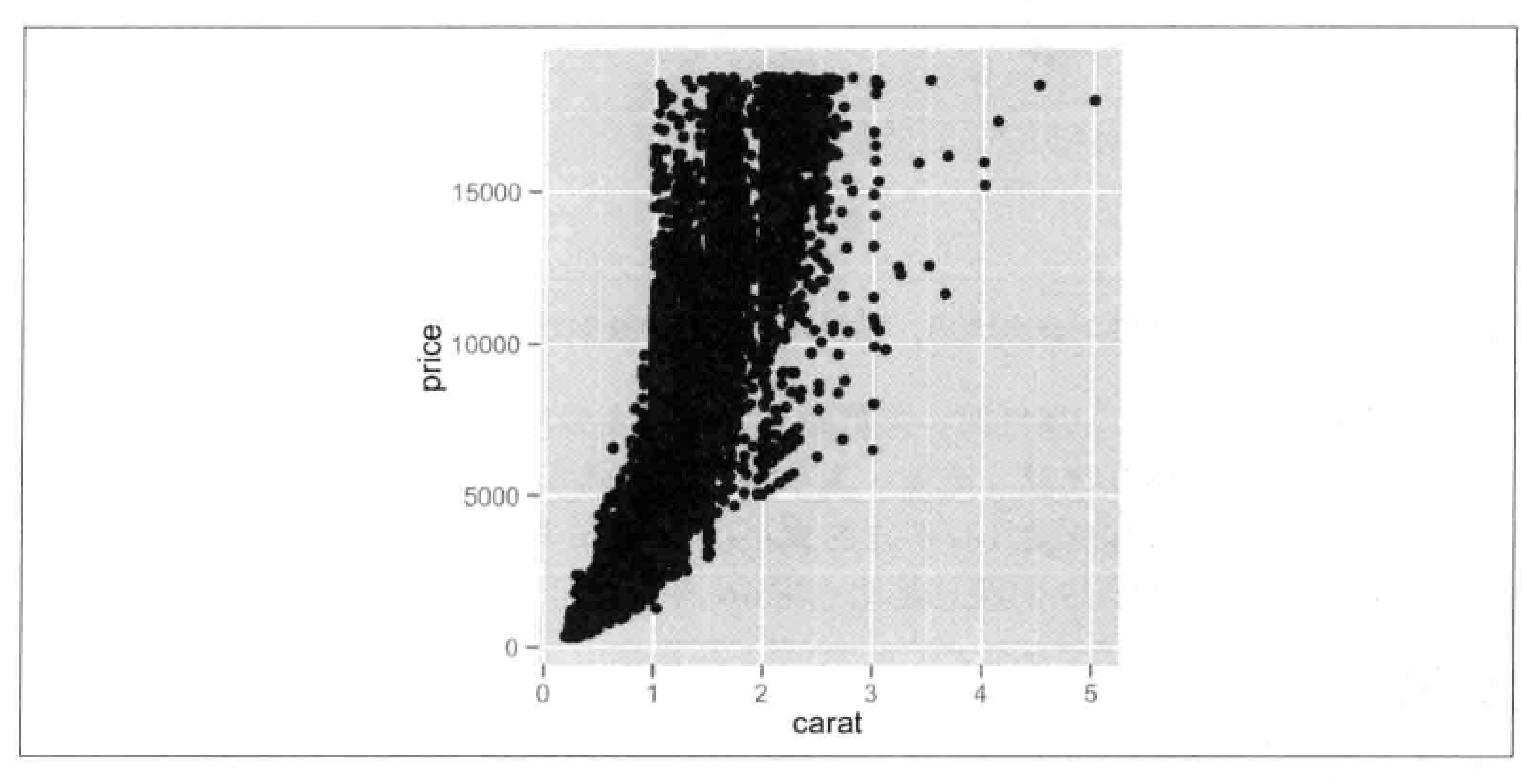


图 5-12　54 000 个数据点的重叠情况

设定 alpha 参数可以使数据点半透明，如图 5-13 所示。通过设定 alpha=.1 和 alpha=.01 使数据点分别具有 90% 和 99% 的透明度：

```
sp + geom_point(alpha=.1)

sp + geom_point(alpha=.01)
```

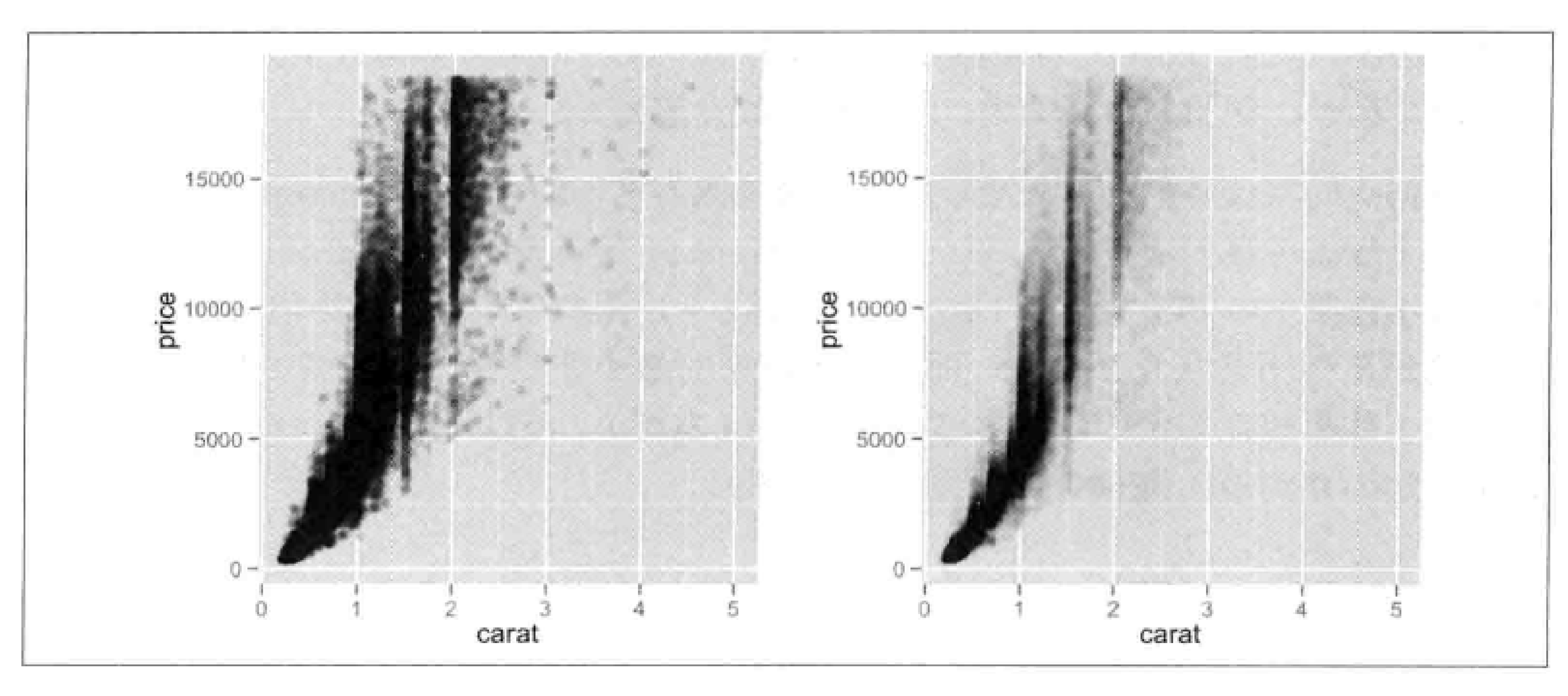


图 5-13　左图：`alpha=.1` 的半透明的散点图　右图：`alpha=.01`

图中，变量 `carat` 的取值为整数和"0.5"的地方有很多垂直带，这意味着人们常按照这些尺寸切割钻石。不过，由于图中数据点过于致密，即使在数据点透明度为 99% 的情况下，图上的大部分区域依然显示为实心的黑色，且数据点的分布情况依然相当模糊。



对于大多数图形，在输出为矢量（如 PDF、EPS 和 SVG）文件时比位图文件（如 TIFF 和 PNG）要小。然而，在数据点非常多时，输出的矢量文件会非常大且渲染过程很慢——这里的具有 99% 透明度的散点图的大小是 1.5 MB！在这种情况下，高精度的位图文件会更小，且在电脑屏幕上的显示速度更快。更多信息可参见本书第 14 章。

另外一个解决方案是将数据点分箱（bin）并以矩形来表示，同时将数据点的密度映射为矩形的填充色，如图 5-14 所示。在分箱绘图情况下，垂直带几乎看不见了。图 5-14 中，左下方的数据点密度更大，这意味着大多数钻石都是比较小且廉价的。

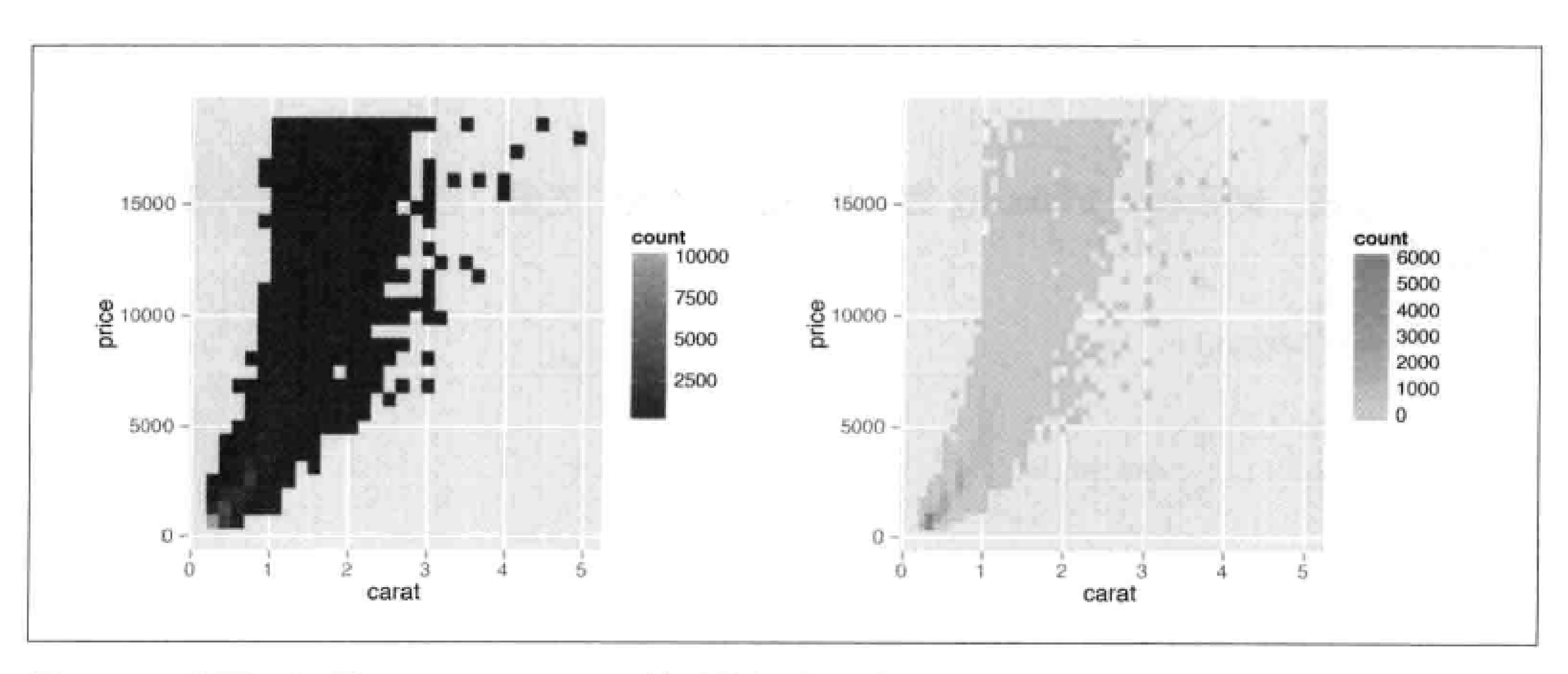


图 5-14　左图：调用 `stat_bin2d()` 函数对数据进行分箱　右图：箱数更多、手动设定数据点颜色及图例的散点图

默认情况下，stat_bin_2d() 函数分别在 x 轴和 y 轴方向上将数据分割为 30 个组，总计 900 个箱子。在第二个版本中，我们将箱数设定为 bin=50。

图中数据点的默认颜色看起来难以区分，这是因为它们的光度（luminosity）变化不大。在第二个版本中我们通过 scale_fill_gradient() 重新设定数据点的颜色，并指定颜色的最小色阶 low 和最大色阶 high。默认情况下，图例中不包括最小值，这是因为颜色标度的范围不是从 0 开始的，而是以各箱中的最小非零值——本例中是 1——为起始点的。如果想在图例中包括零值（见图 5-14 右图），可以调用 limits 参数手动将范围设定为 0 到最大值 6000（见图 5-14 右图）。

```
sp + stat_bin2d()

sp + stat_bin2d(bins=50) +
    scale_fill_gradient(low="lightblue", high="red", limits=c(0, 6000))
```

如果不想将数据分箱并以矩形表示的话，可以调用 stat_binhex() 函数使用六边形代替（见图 5-15）。该函数的工作机制与 stat_bin2d() 类似。使用该函数之前，必须先运行 install.packages("hexbin") 命令安装 hexbin 包。

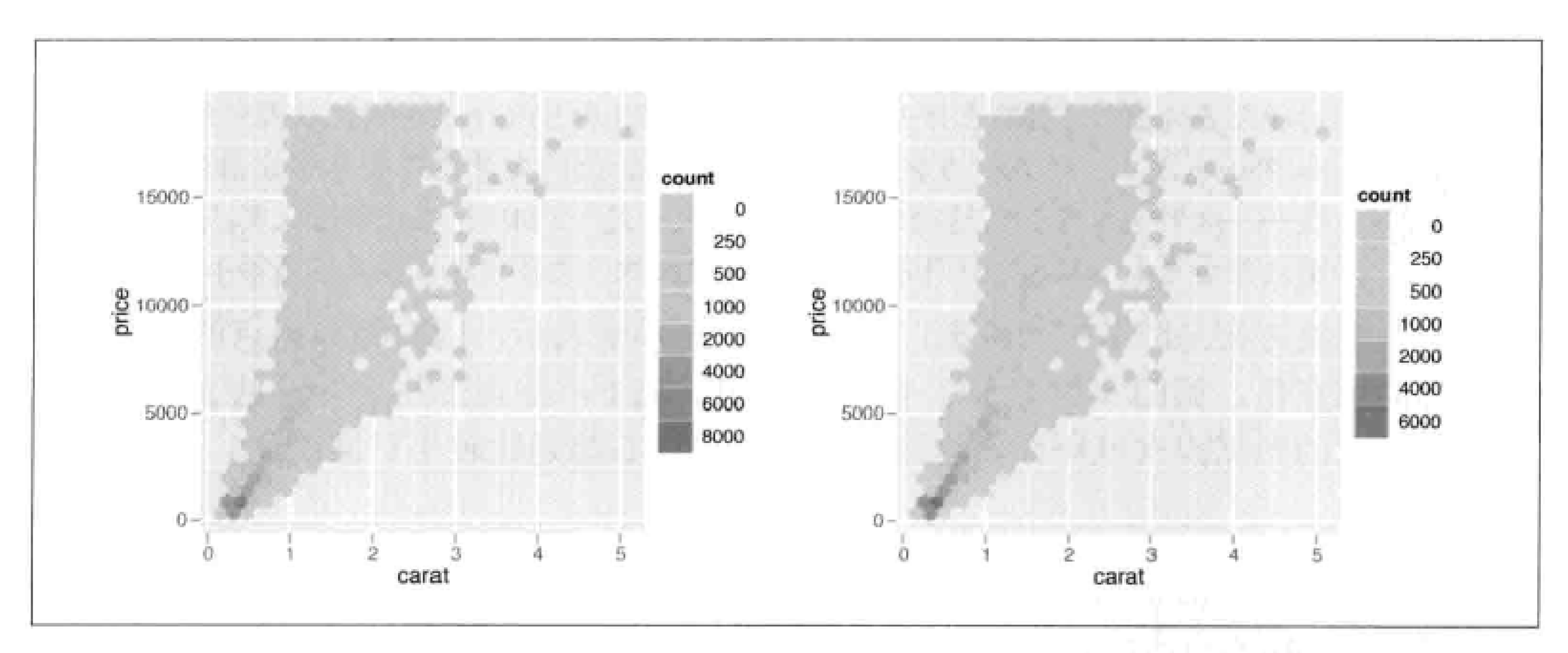


图 5-15　左图：调用 stat_binhex() 函数对数据进行分箱　右图：落在分箱范围外面的格子显示为灰色

```
library(hexbin)

sp + stat_binhex() +
    scale_fill_gradient(low="lightblue", high="red",
                        limits=c(0, 8000))

sp + stat_binhex() +
    scale_fill_gradient(low="lightblue", high="red",
                        breaks=c(0, 250, 500, 1000, 2000, 4000, 6000),
                        limits=c(0, 6000))
```

对于这两种方法，在手动设置分箱范围时，因为数据点太多或者太少，会出现一个落在分箱范围外的箱子，且这个箱子的颜色会显示为灰色，而不是最大值或最小值对应的颜色，如图 5-15 右图所示。

当散点图的其中一个数据轴或者两个数据轴都对应于离散型数据时，也会出现图形重叠的情况，如图 5-16 所示。这时候，可以调用 position_jitter() 函数给数据点增加随机扰动。默认情况下，该函数在每个方向上添加的扰动值为数据点最小精度的 40%，不过，也可以通过 width 和 height 参数对该值进行调整。

```
spl <- ggplot(ChickWeight, aes(x=Time, y=weight))

spl + geom_point()

spl + geom_point(position="jitter")
# 也可以调用 geom_jitter() 函数，两者是等价的

spl + geom_point(position=position_jitter(width=.5, height=0))
```

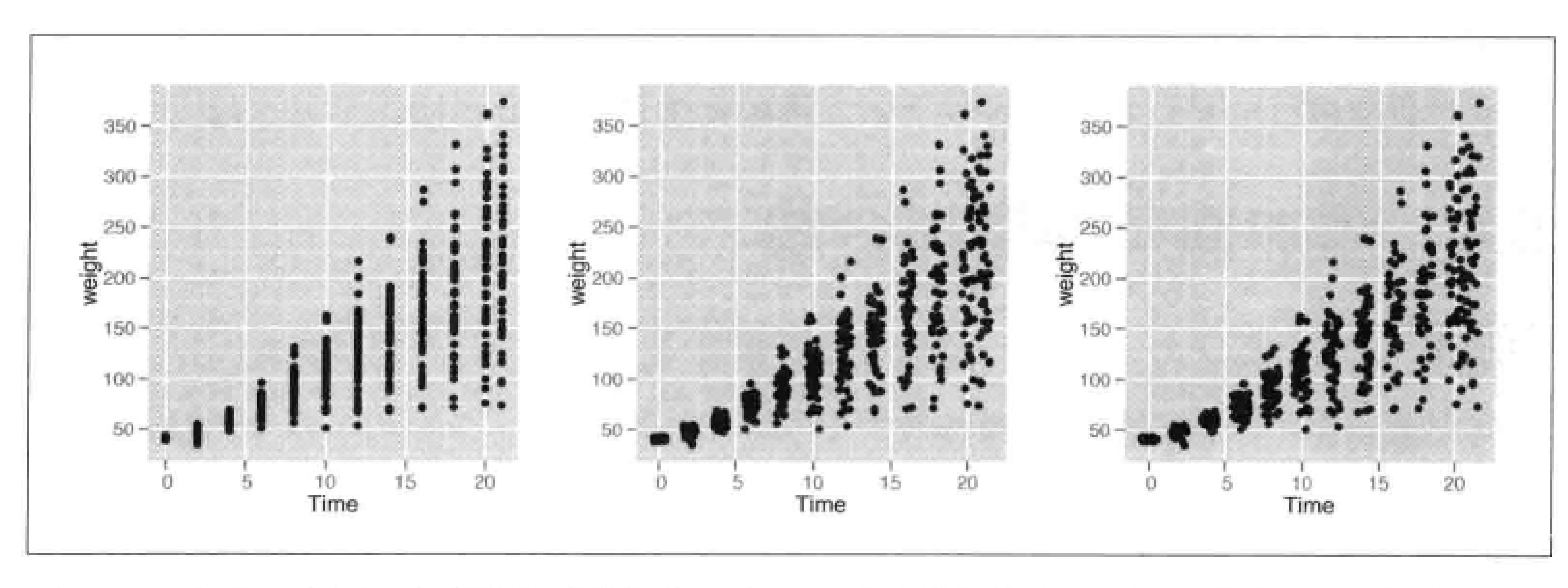


图 5-16　左图：变量 x 为离散型的数据集　中间：添加随机扰动　右图：只在水平方向添加随机扰动

当数据集对应于一个离散型数据轴和一个连续型数据轴时，箱线图可能是一种较好的展示方式，如图 5-17 所示。箱线图所表现的信息与散点图略有不同，因为它很难反映出离散坐标轴上每个位置的数据点数量的信息。箱线图的绘制方式有时候是缺点，但有时候却是恰如其分的可视化方法。

对于 ChickWeights 数据集，其对应的 x 轴本质上是离散的，但其被存储为数值型向量，因此，ggplot() 函数不知如何对该数据集进行分组。如果不告诉 ggplot() 函数如何对数据进行分组，会得到如图 5-17 右图所示的结果。调用 aes(group=...) 可以告诉 ggplot() 如何对数据进行分组。下面，按 Time 变量的取值对数据进行分组：

```
spl + geom_boxplot(aes(group=Time))
```

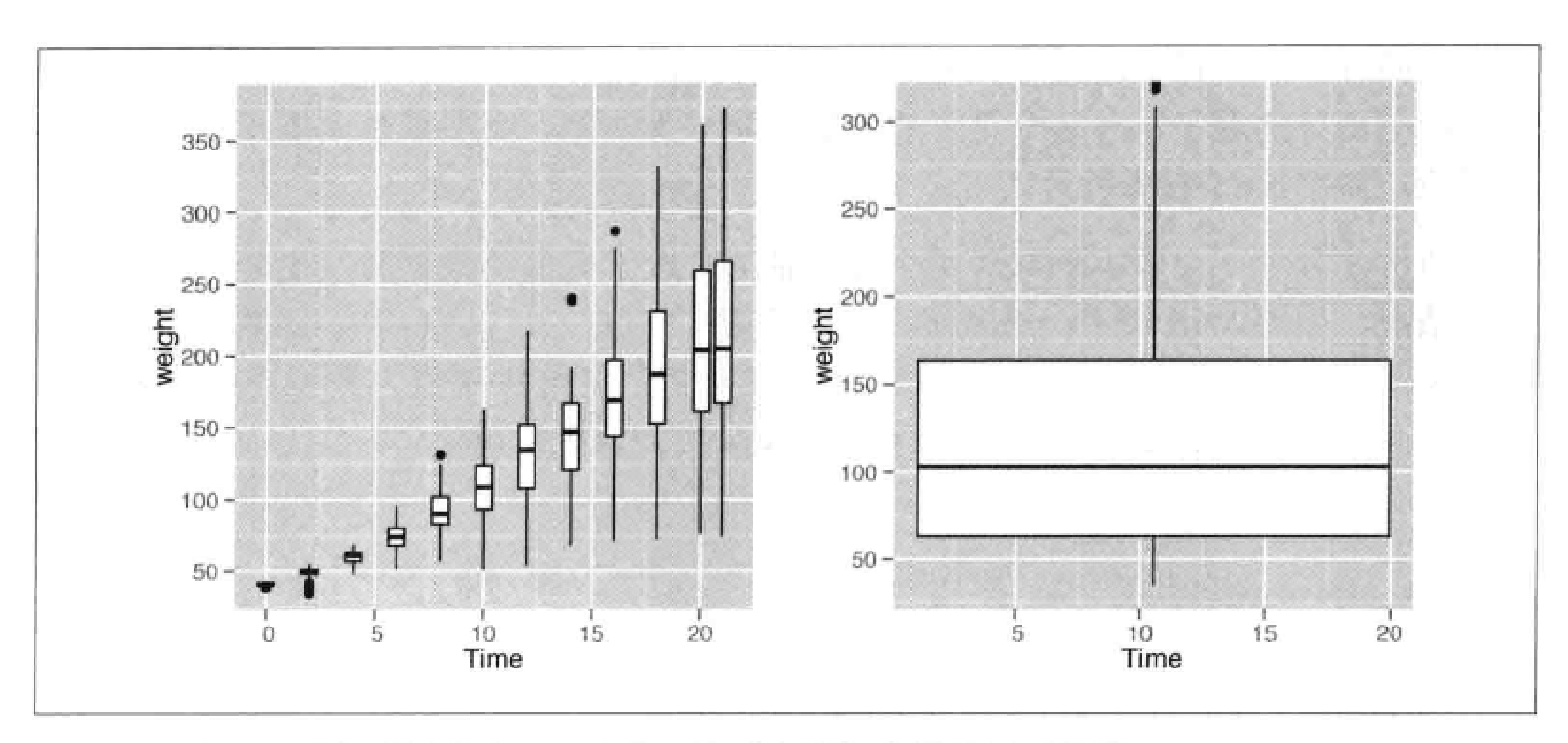


图 5-17　左图：分组绘制箱线图　右图：不设定分组变量的绘图结果

另见

除了对数据进行分箱，我们还可以展示二维的密度估计。具体操作参见 6.12 节。

5.6　添加回归模型拟合线

问题

如何向散点图中添加回归模型拟合线？

方法

运行 `stat_smooth()` 函数并设定 `method=lm` 即可向散点图中添加线性回归拟合线，这将调用 `lm()` 函数对数据拟合线性模型。首先，我们将基本绘图对象存储在对象 `sp` 中，然后，再添加更多的图形部件：

```
library(gcookbook) # 为了使用数据

# 基本绘图对象
sp <- ggplot(heightweight, aes(x=ageYear, y=heightIn))

sp + geom_point() + stat_smooth(method=lm)
```

默认情况下，`stat_smooth()` 函数会为回归拟合线添加 95% 的置信域，置信域对应的置信水平可通过设置 `level` 参数来进行调整。设定参数 `se=FALSE` 时，系统将不会对回归拟合线添加置信域（见图 5-18）：

```
# 99% 置信域
sp + geom_point() + stat_smooth(method=lm, level=0.99)
```

```
# 没有置信域
sp + geom_point() + stat_smooth(method=lm, se=FALSE)
```

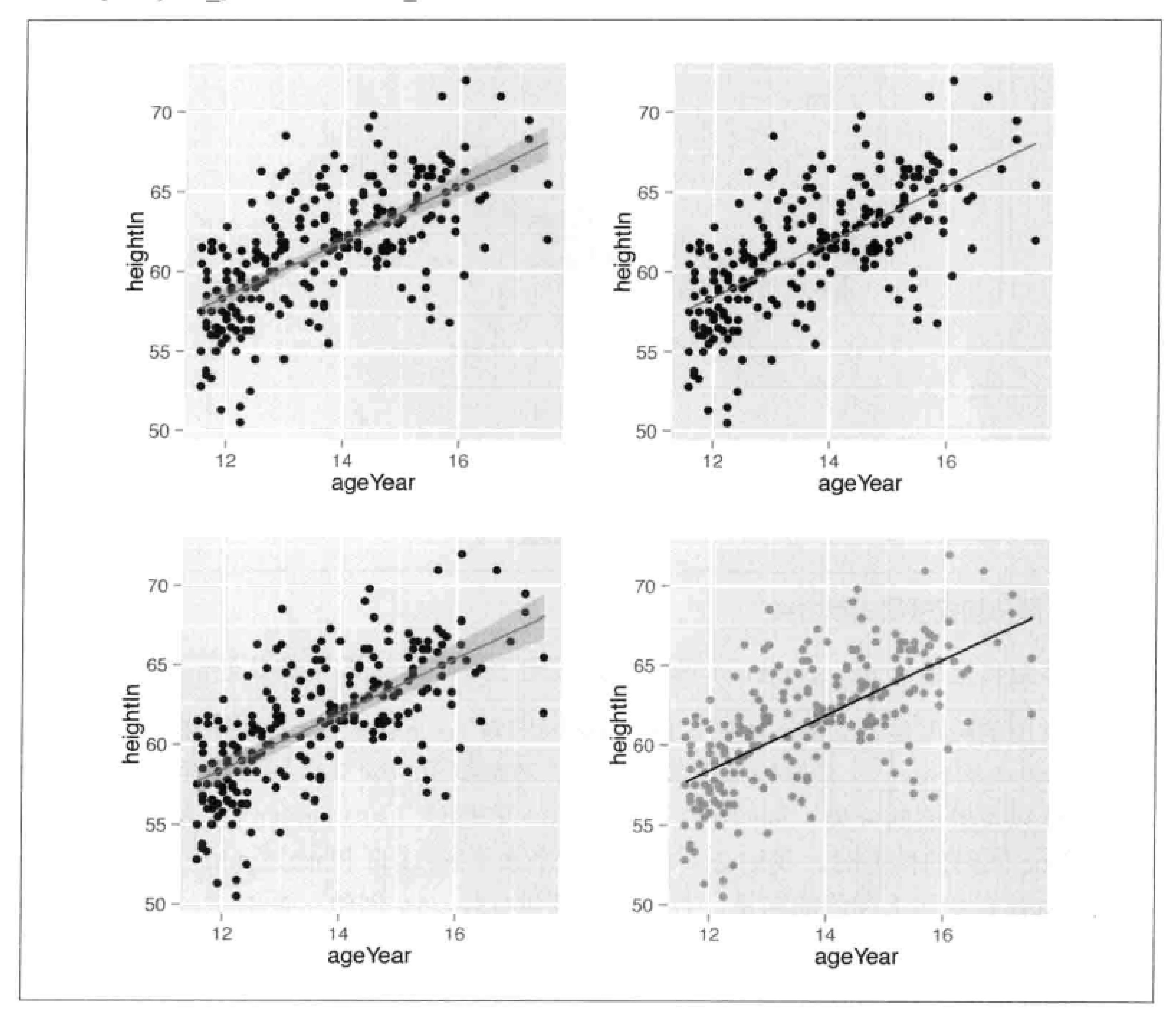


图 5-18　左上：带 95% 置信域的线性拟合线　左下：带 99% 置信域的线性拟合线　右上：没有置信域的线性拟合线　右下：数据点为灰色的黑色线性拟合线

拟合线的默认颜色是蓝色，可以通过设定 colour 参数对其进行调整。与其他直线一样，我们可以对拟合线的线型（linetype）和粗细（size）进行设置。为了突出直线，可设置点的 colour 以使数据点不那么突出（见图 5-18 右下图）：

```
sp + geom_point(colour="grey60") +
    stat_smooth(method=lm, se=FALSE, colour="black")
```

讨论

线性模型并不是唯一可对数据进行拟合的模型——事实上，它甚至不是默认的模型。如果在调用 stat_smooth() 函数时未指定模型类型，那么，该函数会对数据拟合 loess 曲线（局部加权多项式），如图 5-19 所示。下面两行命令的输出结果相同：

```
sp + geom_point(colour="grey60") + stat_smooth()
sp + geom_point(colour="grey60") + stat_smooth(method=loess)
```

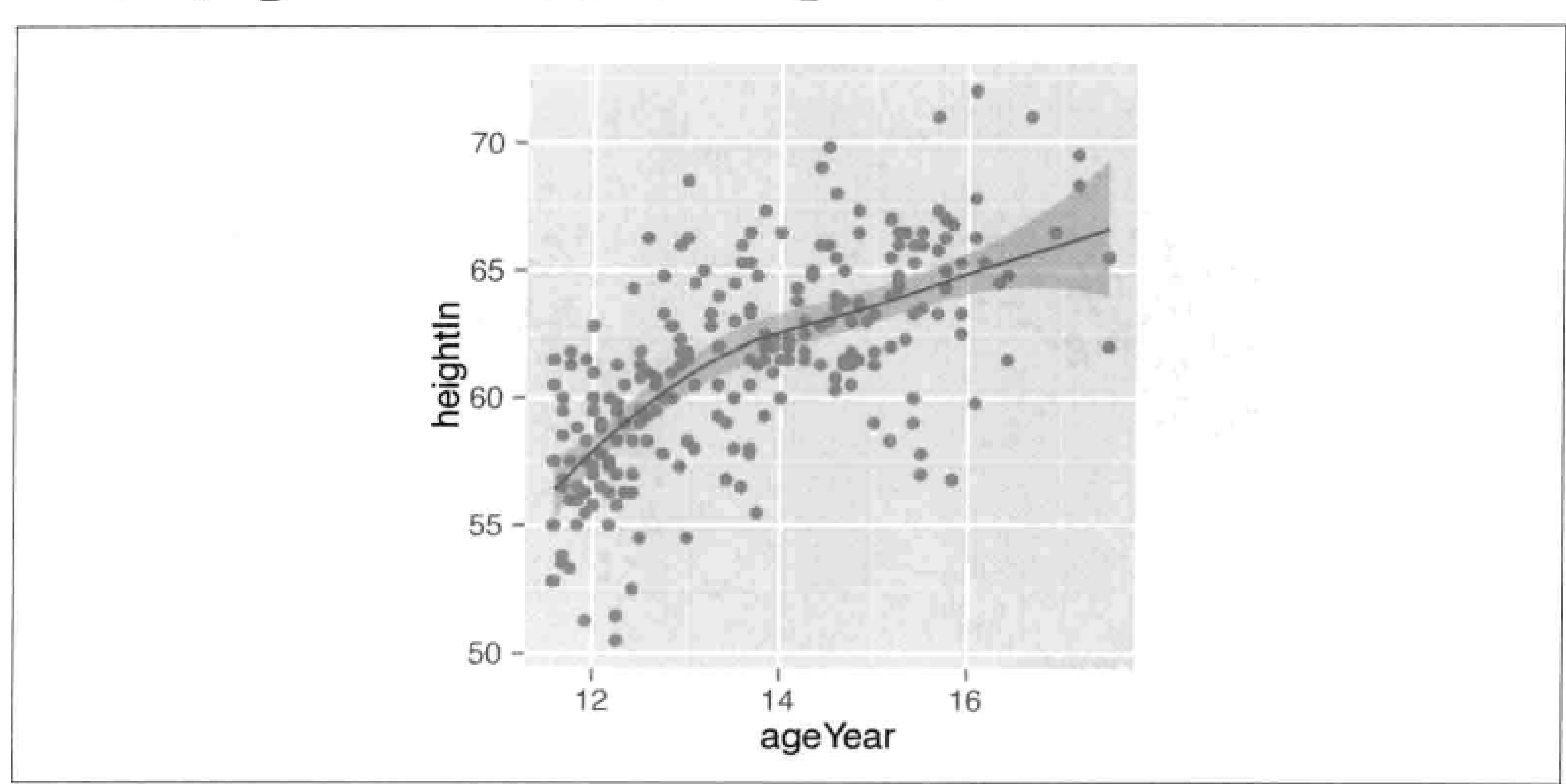


图 5-19　局部加权多项式拟合线

通过将参数传递给 `stat_smooth()` 函数可以设定 `loess()` 函数中的其他附加参数。

另一种常用的模型是 Logistic 回归。Logistic 回归对 `heightweight` 数据集不适用，但对 `MASS` 包中的 `biopsy` 数据集拟合效果良好。该数据集包含 9 个与乳腺癌活检组织相关的指标以及肿瘤的分类，包括良性（`benign`）和恶性（`malignant`）两种。在预处理 Logistic 回归的数据时，我们必须将具有两个水平 `benign` 和 `malignant` 的因子型变量转化为具有 0 和 1 取值的向量。首先，为变量 `biopsy` 创建一个副本，接着将其重编码为数值型的变量并存储在列 `classn` 中：

```
library(MASS) # 为了使用数据

b <- biopsy

b$classn[b$class=="benign"]    <- 0
b$classn[b$class=="malignant"] <- 1
b

      ID V1 V2 V3 V4 V5 V6 V7 V8 V9     class classn
 1000025  5  1  1  1  2  1  3  1  1    benign      0
 1002945  5  4  4  5  7 10  3  2  1    benign      0
 1015425  3  1  1  1  2  2  3  1  1    benign      0
...
  897471  4  8  6  4  3  4 10  6  1 malignant      1
  897471  4  8  8  5  4  5 10  4  1 malignant      1
```

虽然，涉及的指标很多，但在本例中，我们只探寻变量 `V1`（肿块厚度）和肿瘤类型的关系。图中数据点重叠程度严重，因此，需要向数据点添加一些扰动，同时，将数据

点设置为半透明（alpha=.4）、点形设置为空心圆（shape=21），并使用略小的数据点（size=1.5）。令 stat_smooth() 函数使用选项为 family=binomial 的 glm() 函数向散点图添加 Logistic 回归拟合线（见图 5-20）：

```
ggplot(b, aes(x=V1, y=classn)) +
    geom_point(position=position_jitter(width=0.3, height=0.06), alpha=0.4,
               shape=21, size=1.5) +
    stat_smooth(method=glm, family=binomial)
```

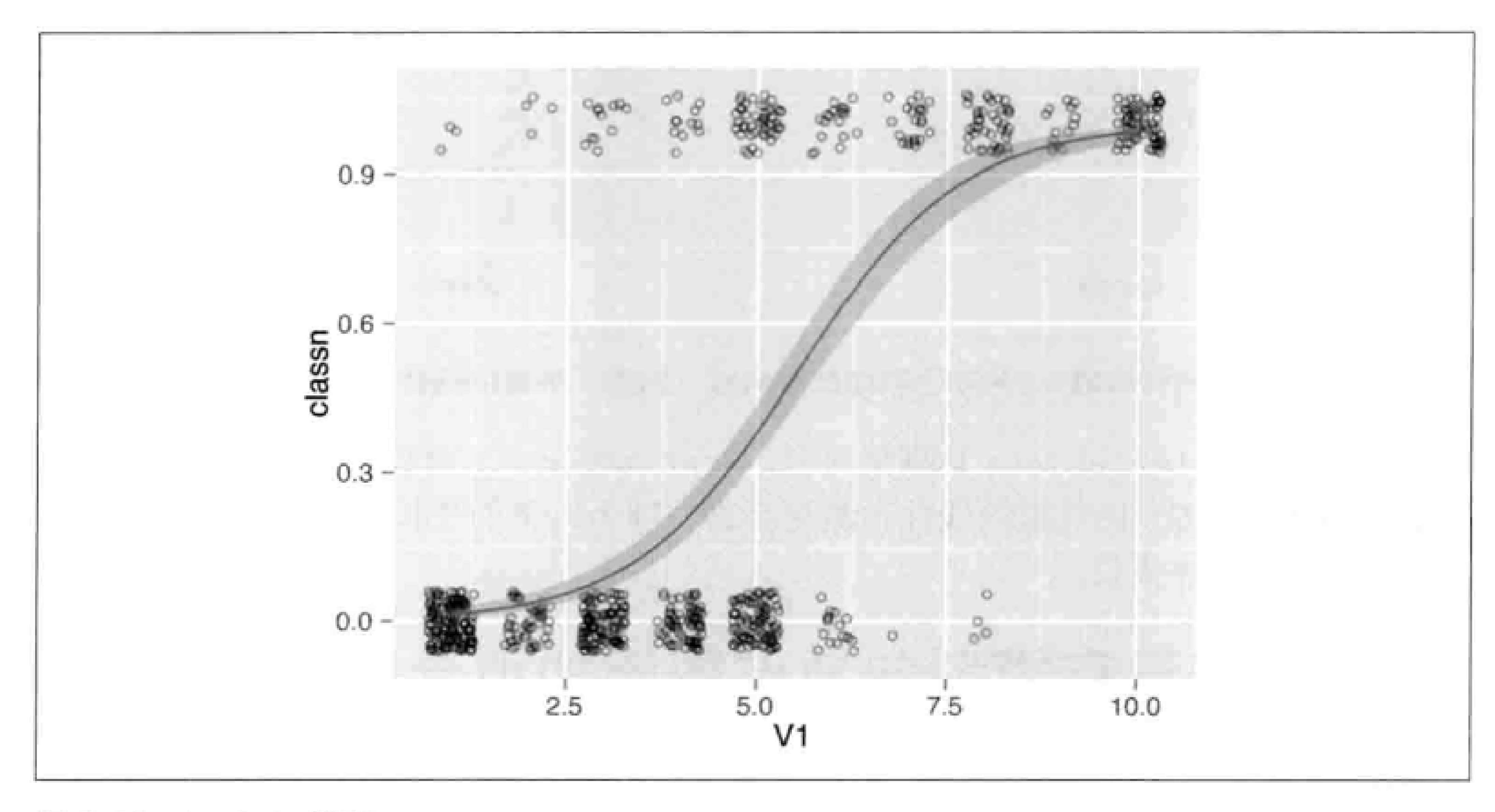


图 5-20　Logistic 模型

如果散点图对应的数据集按照某个因子型变量进行了分组，且已将分组变量映射给 colour 和 shape 属性，上述命令将针对各个组分别绘制模型拟合线。首先，创建一个基本绘图对象 sps；然后，向其添加 loess 线；最后，将点的颜色设定为半透明（alpha=.4）以弱化数据点的显示（见图 5-21）。

```
sps <- ggplot(heightweight, aes(x=ageYear, y=heightIn, colour=sex)) +
       geom_point() +
       scale_colour_brewer(palette="Set1")

sps + geom_smooth()
```

注意，男性分组的蓝色拟合线并没有绘制到图形的右边界。其中有两个原因：第一个原因在于，默认情况下 stat_smooth 函数将预测值的范围限定在预测数据对应的范围内（对应于 x 轴）；第二个原因在于，即使对模型进行外推，loess() 函数也只能根据整组数据对应的 x 轴的范围进行预测。

如果想基于数据集对拟合线进行外推，如图 5-21 右图所示，必须保证绘图过程中能够调用的是支持外推的模型，比如 lm()，并将选项 fullrange=TRUE 传递给 stat_smooth() 函数：

```
sps + geom_smooth(method=lm, se=FALSE, fullrange=TRUE)
```

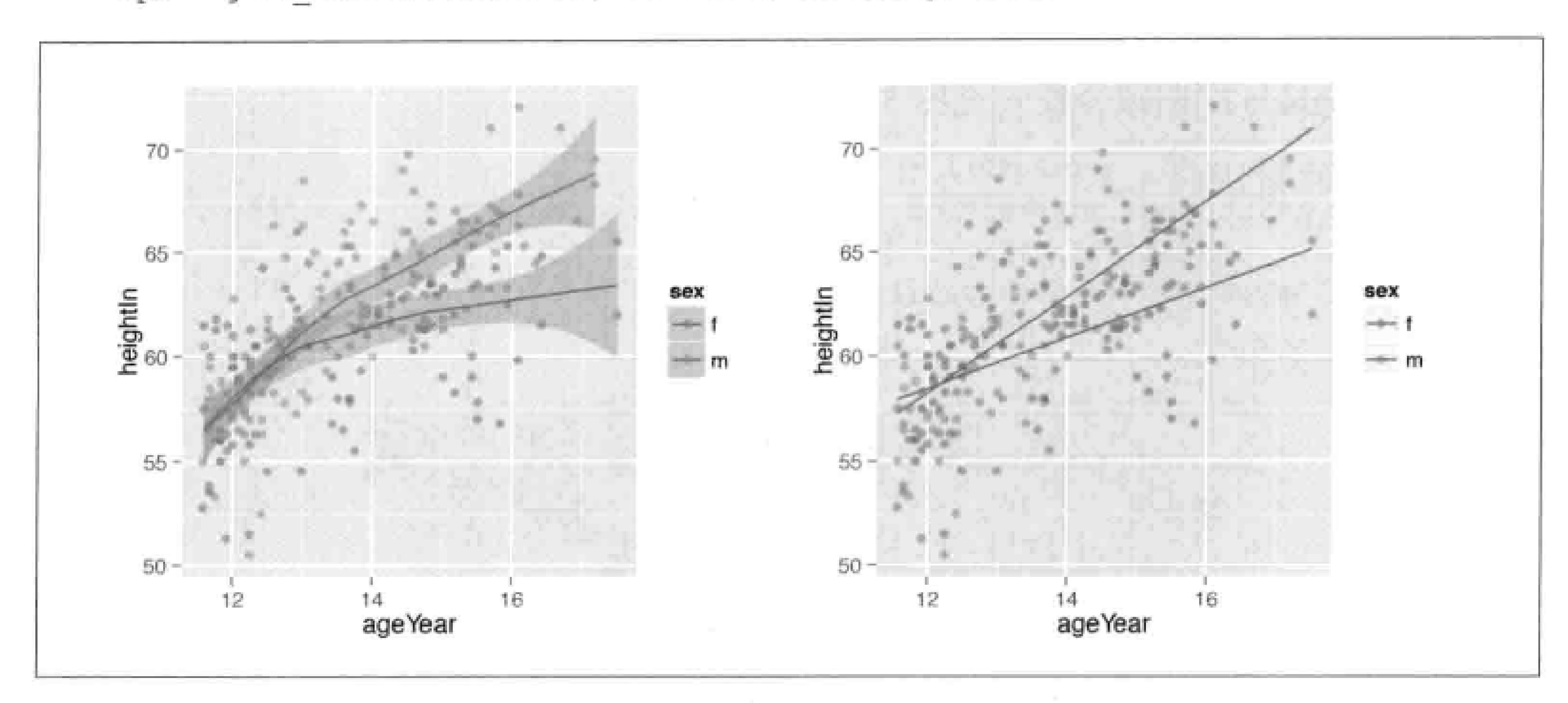


图 5-21　左图：针对各组数据分别绘制模型拟合线　右图：外推的线性拟合线

对于本例中的 heightweight 数据集而言，stat_smooth() 函数默认的方法（不带外推的 LOESS 方法）比外推的线性预测更合理，因为人类的生长过程并非线性的，且人类也不会一直在生长。

5.7　根据已有模型向散点图添加拟合线

问题

对数据集建立回归模型之后，如何将模型对应的拟合线添加到散点图上？

方法

常用的向散点图添加模型拟合线的方法是调用 stat_smooth() 函数，这在 5.6 节中已有所介绍。然而，有时候我们想要建立自己的模型，再将模型拟合线添加到散点图上，这样做可以使所用的模型与图中所见保持一致。

本例中，我们使用 lm() 函数建立一个以 ageYear 为预测变量对 heightIn 进行预测的模型。然后，调用 predict() 函数计算预测变量 ageYear 各取值所对应的 heightIn 变量的预测值：

```
library(gcookbook) # 为了使用数据

model <- lm(heightIn ~ ageYear + I(ageYear^2), heightweight)
model

Call:
lm(formula = heightIn ~ ageYear + I(ageYear^2), data = heightweight)
```

```
Coefficients:
 (Intercept)      ageYear  I(ageYear^2)
    -10.3136       8.6673       -0.2478

# 创建一个包含变量 ageYear 的列，并对其进行插值
xmin <- min(heightweight$ageYear)
xmax <- max(heightweight$ageYear)
predicted <- data.frame(ageYear=seq(xmin, xmax, length.out=100))

# 计算变量 heightIn 的预测值
predicted$heightIn <- predict(model, predicted)
predicted

  ageYear heightIn
 11.5800 56.82624
 11.6398 57.00047
 ...
 17.4402 65.47875
 17.5000 65.47933
```

现在，我们可以将数据点和模型预测值一起绘制在图形上（见图 5-22）：

```
sp <- ggplot(heightweight, aes(x=ageYear, y=heightIn)) +
      geom_point(colour="grey40")

sp + geom_line(data=predicted, size=1)
```

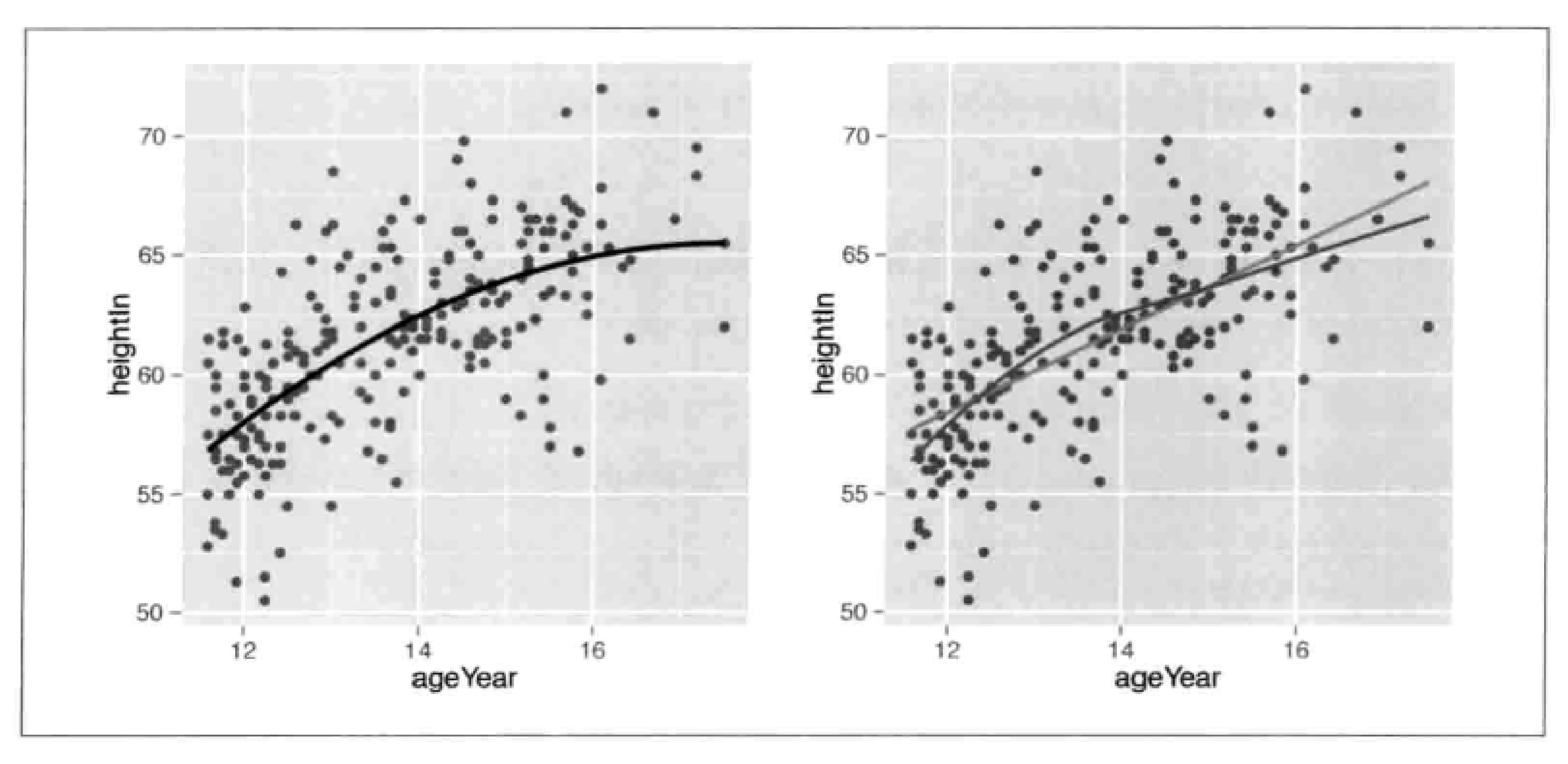


图 5-22　左图：基于 lm 对象的二次预测曲线　右图：基于线性模型（红线）和 LOESS 模型（蓝线）的拟合线

讨论

无论哪种模型，只要有对应的 predict() 方法，则其都可用来绘制拟合线。举个例子：

lm() 函数和 loess() 函数对应的 predict() 方法分别是 predict.lm() 和 predict.loess() 等，因此，这两个模型都可以用来绘制模型拟合线。

应用下面定义的 predictvals() 函数可以简化向散点图添加模型拟合线的过程。使用时，只需向其传递一个模型作为参数，该函数就会自动查询变量名、预测变量范围、并返回一个包含预测变量和模型预测值的数据框。将该数据框传递给 geom_line() 函数即可绘制我们在前面看到的模型拟合线：

```
# 根据模型和变量 xvar 预测变量 yvar
# 仅支持单一预测变量的模型
# xrange: x 轴范围，当值为 NULL 时，等于模型对象中提取的 x 轴范围；当设定为包含两个数字的
# 向量时，两个数字分别对应于 x 轴范围的上下限
# sample:x 轴上包含的样本数量
# ...: 可传递给 predict() 函数的其他参数
predictvals <- function(model, xvar, yvar, xrange=NULL, samples=100, ...) {

  # 如果 xrange 没有输入，则从模型对象中自动提取 x 轴范围作为参数
  # 提取 xrange 参数的方法视模型而定
  if (is.null(xrange)) {
    if (any(class(model) %in% c("lm","glm")))
      xrange <- range(model$model[[xvar]])
    else if (any(class(model) %in% "loess"))
      xrange <- range(model$x)
  }

  newdata <- data.frame(x = seq(xrange[1], xrange[2], length.out = samples))
  names(newdata) <- xvar
  newdata[[yvar]] <- predict(model, newdata = newdata, ...)
  newdata
}
```

调用 lm() 函数和 loess() 函数对数据集建立线性模型和 LOESS 模型（见图 5-22）：

```
modlinear <- lm(heightIn ~ ageYear, heightweight)

modloess <- loess(heightIn ~ ageYear, heightweight)
```

针对两个模型分别调用 predictvals() 函数，并将得到的结果（数据框）传递给 geom_line()：

```
lm_predicted    <- predictvals(modlinear, "ageYear", "heightIn")
loess_predicted <- predictvals(modloess, "ageYear", "heightIn")

sp + geom_line(data=lm_predicted, colour="red", size=.8) +
     geom_line(data=loess_predicted, colour="blue", size=.8)
```

对于具有非线性连接函数的 glm 模型，需要将 predictvals() 函数的参数设定为 type="response"。这样做的原因在于，默认情况下该函数返回的预测结果是基于线性项的，而不是基于响应变量（y）的。

以 MASS 包中的 biopsy 数据为例演示一下上述过程。与 5.6 节中一样，下面用变量 V1 来预测变量 class。Logistic 模型对应的值须是介于 0 到 1 之间的数值，这里变量

class 是因子型变量，因而，要先将变量 class 的取值转化为 0 和 1。

```
library(MASS) # 为了使用数据
b <- biopsy

b$classn[b$class=="benign"]    <- 0
b$classn[b$class=="malignant"] <- 1
```

下面，建立 Logistic 回归模型：

```
fitlogistic <- glm(classn ~ V1, b, family=binomial)
```

最后，绘制带扰动和 fitlogistic 线的散点图。通过指定 RGB 颜色值将直线设定为蓝色，同时，指定参数 size=1 使线条更宽（见图 5-23）。

```
# 获取预测值
glm_predicted <- predictvals(fitlogistic, "V1", "classn", type="response")

ggplot(b, aes(x=V1, y=classn)) +
    geom_point(position=position_jitter(width=.3, height=.08), alpha=0.4,
               shape=21, size=1.5) +
    geom_line(data=glm_predicted, colour="#1177FF", size=1)
```

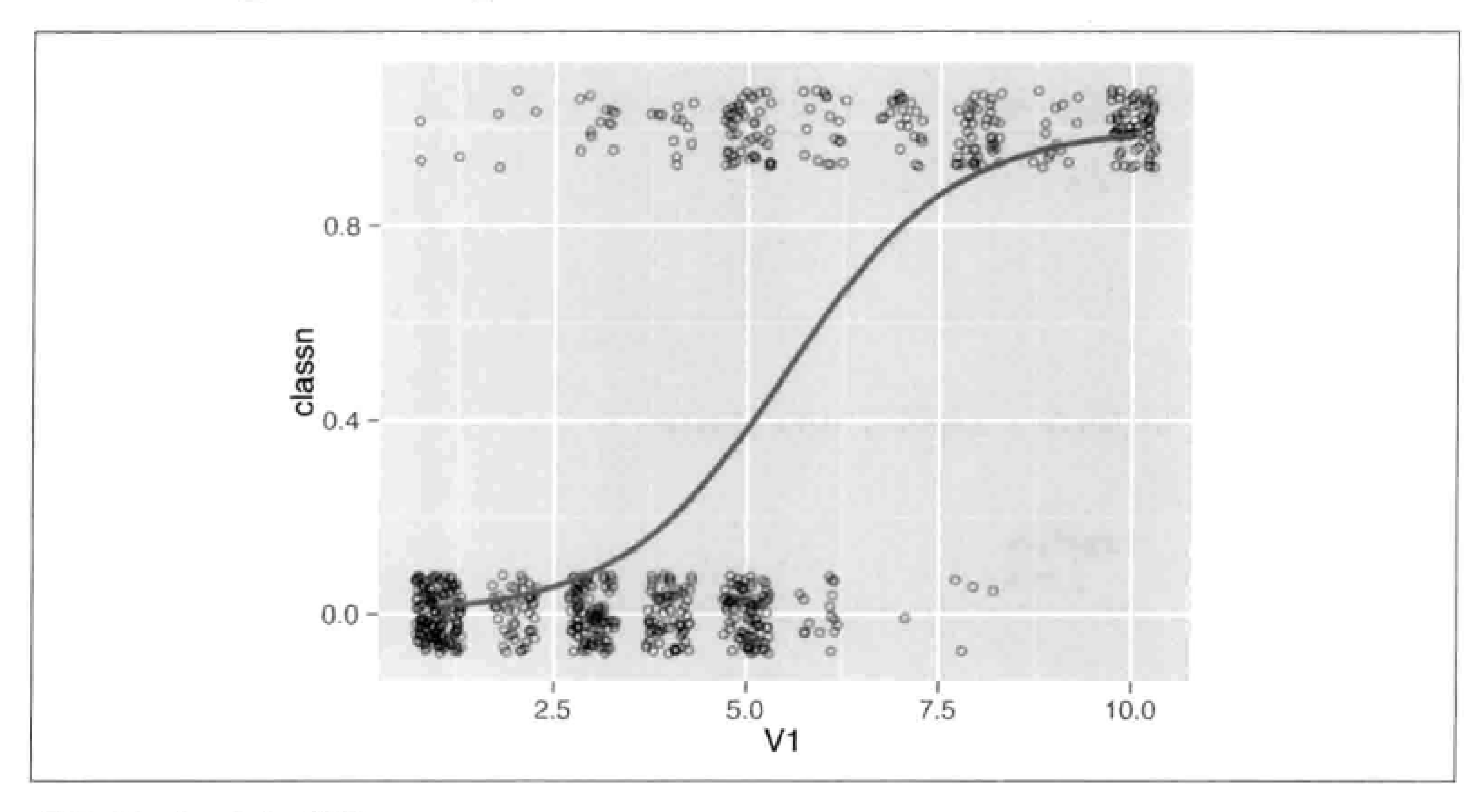


图 5-23　Logistic 模型

5.8　添加来自多个模型的拟合线

问题

对数据集建立了回归模型之后，如何绘制模型对应的拟合线？

方法

使用上文提到的 predictvals() 函数和来自 **plyr** 包的 dlply() 及 ldply() 函数即可。

根据变量 sex 的水平对 heightweight 数据集进行分组，调用 lm() 函数对每组数据分别建立线性模型，并将模型结果存放在一个列表内。随后，通过下面定义的 make_model() 函数建立模型。调用该函数时，向其输入一个数据框作为参数，该函数会返回一个 lm 对象。也可以根据数据集自定义模型：

```
make_model <- function(data) {
    lm(heightIn ~ ageYear, data)
}
```

有了上面的函数之后，可以调用 dlply() 函数分别针对数据集的各个子集建立模型。在执行过程中，函数会根据分组变量 sex 将数据框切分为不同的子集，并对各个子集执行 make.model() 函数。本例中，heightweight 数据集被切分为男性组和女性组，make_model() 函数分别对两个组的数据建立模型。调用 dlply() 函数将模型结果输出到列表中，并返回列表：

```
library(gcookbook) # 为了使用数据
library(plyr)
models <- dlply(heightweight, "sex", .fun = make_model)

# 打印出两个 lm 对象 f 和 m 组成的列表
models

$f

Call:
lm(formula = heightIn ~ ageYear, data = data)

Coefficients:
(Intercept)      ageYear
     43.963        1.209


$m

Call:
lm(formula = heightIn ~ ageYear, data = data)

Coefficients:
(Intercept)      ageYear
     30.658        2.301

attr(,"split_type")
[1] "data.frame"
attr(,"split_labels")
  sex
```

```
1   f
2   m
```

得到模型对象之后，配合使用 `ldply()` 函数和 `predictvals()` 函数即可获取两个模型对应的预测值。

```
predvals <- ldply(models, .fun=predictvals, xvar="ageYear", yvar="heightIn")
predvals ##

 sex  ageYear heightIn
   f 11.58000 57.96250
   f 11.63980 58.03478
   f 11.69960 58.10707
...
   m 17.38040 70.64912
   m 17.44020 70.78671
   m 17.50000 70.92430
```

最后，绘制带预测值的散点图（见图 5-24）：

```
ggplot(heightweight, aes(x=ageYear, y=heightIn, colour=sex)) +
    geom_point() + geom_line(data=predvals
```

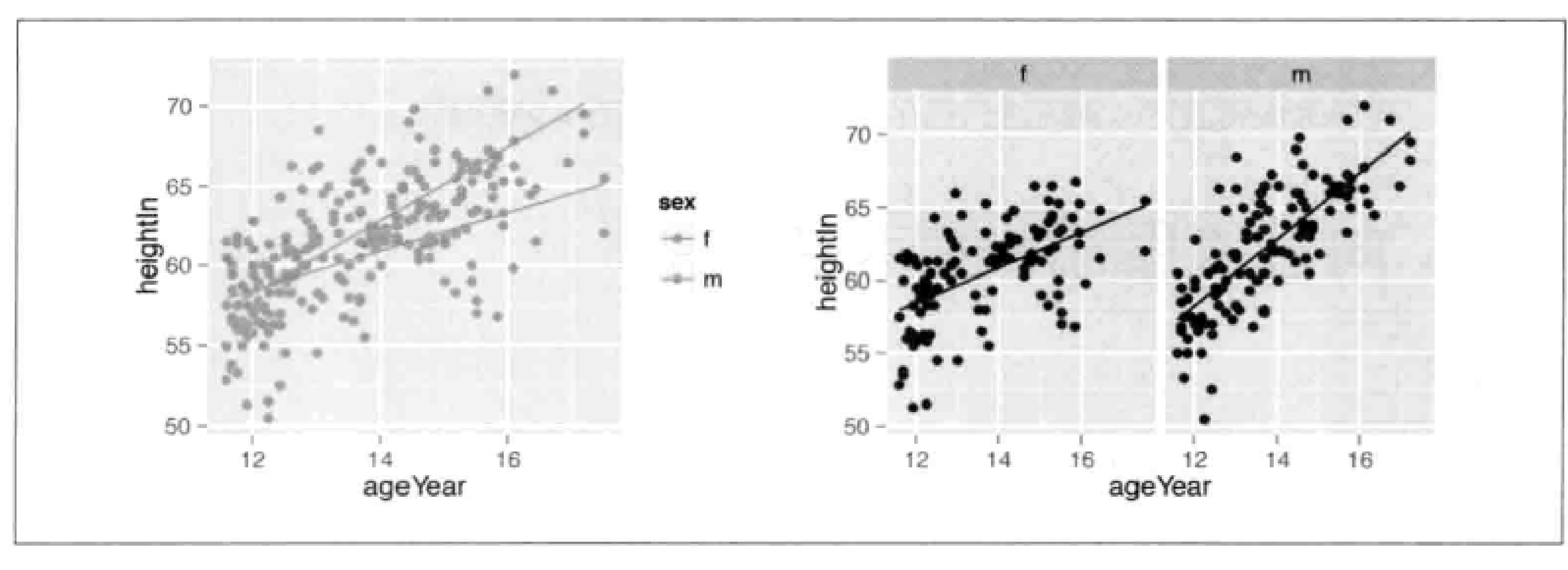


图 5-24　左图：基于两个独立的 lm 对象的预测值，两个对象各自对应于数据的一个子集　右图：分面绘图

讨论

`dlply()` 函数和 `ldply()` 函数的作用是切分数据，对各个部分执行某一函数，并对执行结果进行重组。

在前面的代码中，每组数据的预测值对应的 x 轴的范围与每组数据所对应的 x 轴的范围相同，并未向外延伸。男性组的预测值终止于男性组中年龄最大的点；女性组的预测值更靠右，终止于女性组中年龄最大的点。为了使两组预测线对应的 x 轴范围与整个数据集对应的 x 轴范围相同，可以像下面这样向其传递一个 `xrange` 参数：

```
predvals <- ldply(models, .fun=predictvals, xvar="ageYear", yvar="heightIn",
                  xrange=range(heightweight$ageYear))
```

接下来的绘图操作与前面类似。

```
ggplot(heightweight, aes(x=ageYear, y=heightIn, colour=sex)) +
    geom_point() + geom_line(data=predvals)
```

从图 5-25 可以看到，男性组的模型拟合线延伸到了女性组的右边界。谨记一点：外推拟合线并非总是适用的，其适用与否要视数据特性及模型假设而定。

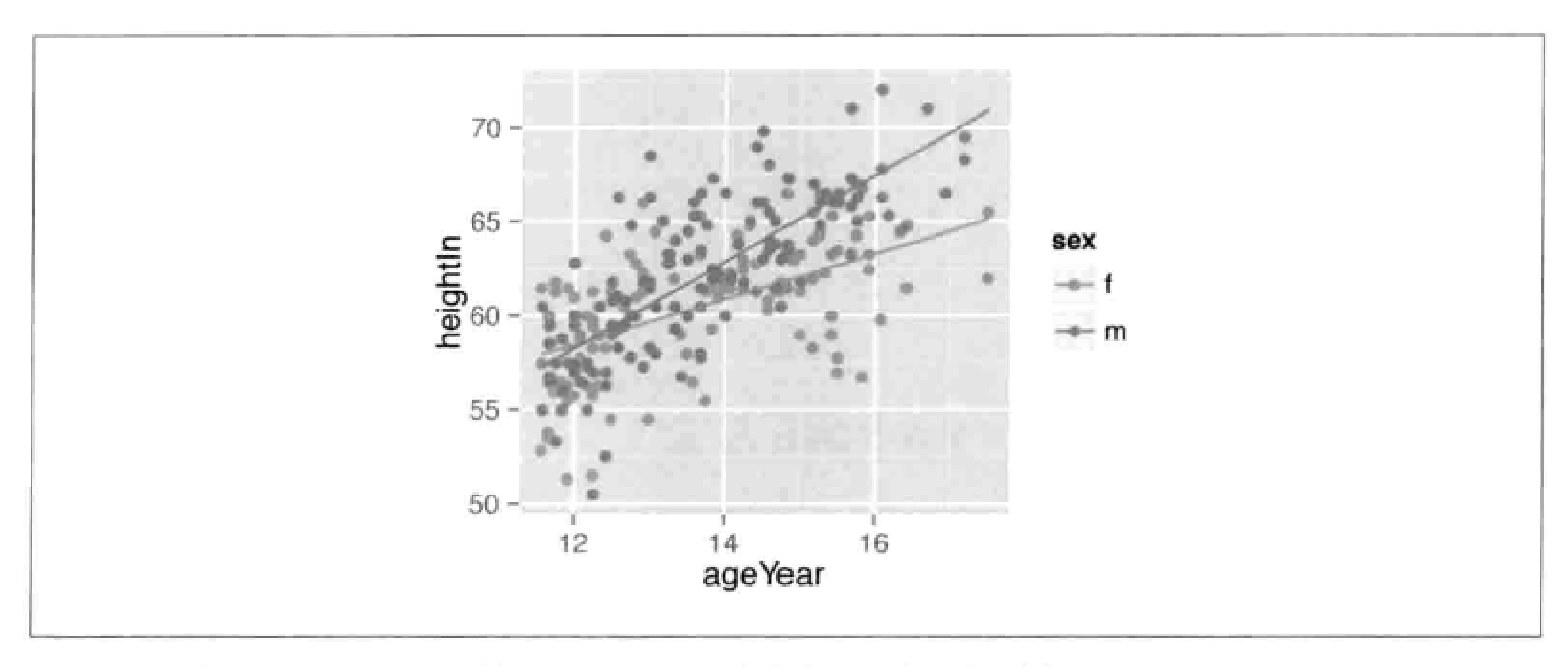


图 5-25　将每组预测值的 x 轴范围外推至整个数据集对应的 x 轴范围

5.9　向散点图添加模型系数

问题

如何向图形添加模型的数值信息？

方法

简单的文本以注释形式添加到图形上面即可。下面的例子中，我们会建立一个线性模型，并调用 5.7 节中定义的 predictval() 函数创建一条预测线。最后，向图形添加注释。

```
library(gcookbook) # 为了使用数据

model <- lm(heightIn ~ ageYear, heightweight)
summary(model)

Call:
lm(formula = heightIn ~ ageYear, data = heightweight)

Residuals:
    Min      1Q  Median      3Q     Max
-8.3517 -1.9006  0.1378  1.9071  8.3371

Coefficients:
```

```
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 37.4356     1.8281   20.48   <2e-16 ***
ageYear      1.7483     0.1329   13.15   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.989 on 234 degrees of freedom
Multiple R-squared: 0.4249,  Adjusted R-squared: 0.4225
F-statistic: 172.9 on 1 and 234 DF,  p-value: < 2.2e-16
```

上面的结果表明模型的 r^2 值是 0.4249。我们创建一个图形，并调用 annotate() 函数向其手动添加文本（见图 5-26）：

```
# 首先，生成预测值
pred <- predictvals(model, "ageYear", "heightIn")
sp <- ggplot(heightweight, aes(x=ageYear, y=heightIn)) + geom_point() +
    geom_line(data=pred)

sp + annotate("text", label="r^2=0.42", x=16.5, y=52)
```

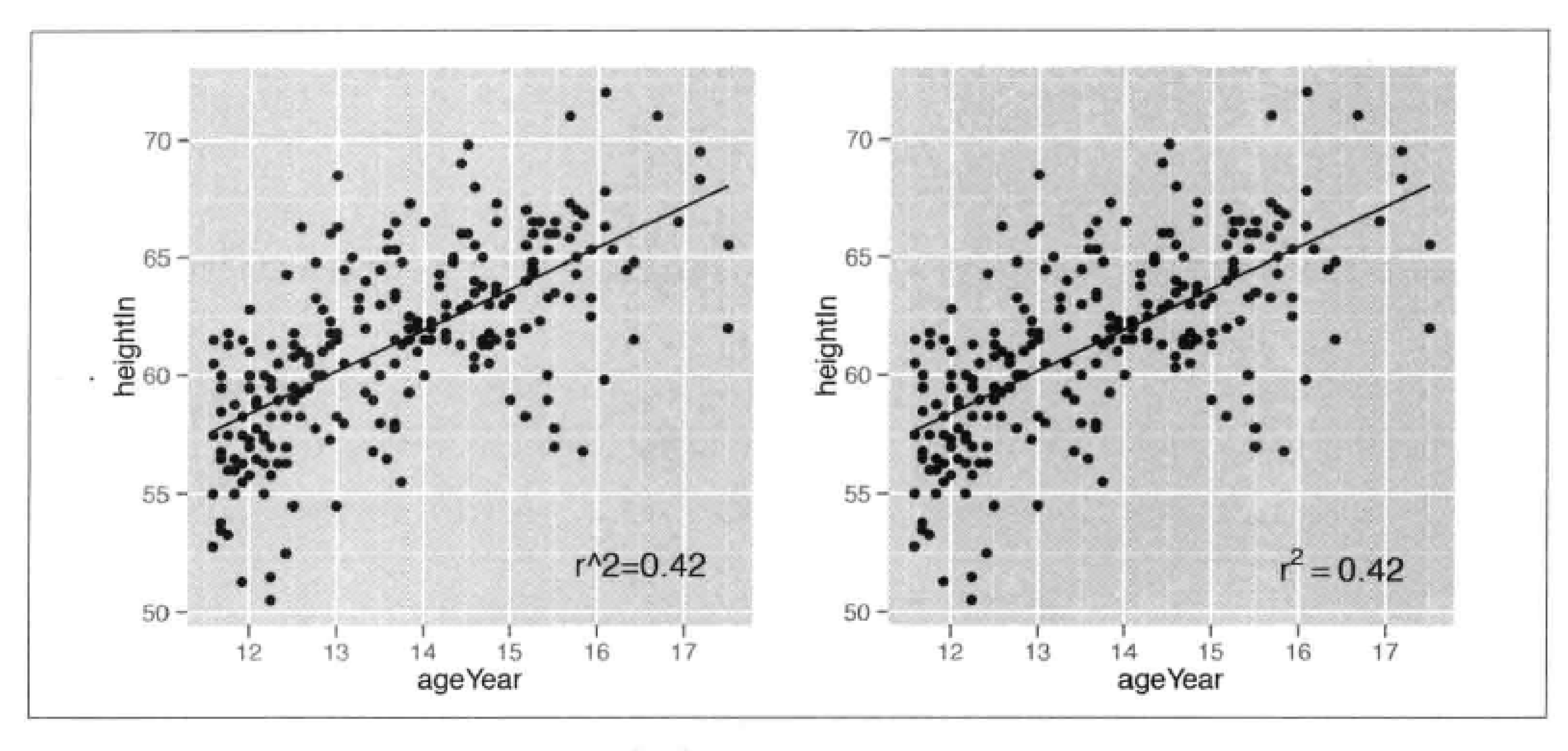


图 5-26　左图：纯文本　右图：数学公式

如果不想使用纯文本字符串当注释的话，可以通过设置 parse=TRUE 调用 R 的数学表达式语法来输入公式。

```
sp + annotate("text", label="r^2 == 0.42", parse = TRUE, x=16.5, y=52)
```

讨论

ggplot2 中的文本对象不能直接以表达式对象作为输入，其参数通常是一个字符串，接收字符串后，通过 parse(text="a + b") 函数将其转化为公式。

使用数学公式作为注释时，必须使用正确的语法才能保证系统输出一个合法的 R 表达式对象。把公式封装在 expression() 内部，检验其输出结果可以辅助判断 R 表达式的有效性

（确保公式两边没有引号）。本例中 == 是公式中表达等号的合法字符，而 = 则是非法字符。

```
expression(r^2 == 0.42) # 合法公式

expression(r^2 == 0.42)

expression(r^2 = 0.42) # 非法公式

Error: unexpected '=' in "expression(r^2 ="
```

还可以自动提取模型对象的值并创建一个引用这些值的公式。在接下来的例子中，我们创建一个字符串，对其进行解析后，会返回一个合法的公式：

```
eqn <- as.character(as.expression(
  substitute(italic(y) == a + b * italic(x) * "," ~~ italic(r)^2 ~ "=" ~ r2,
    list(a = format(coef(model)[1], digits=3),
         b = format(coef(model)[2], digits=3),
        r2 = format(summary(model)$r.squared, digits=2)
       ))))
eqn

"italic(y) == \"37.4\" + \"1.75\" * italic(x) * \",\" ~ ~italic(r)^2 ~ \"=\" ~
\"0.42\""

parse(text=eqn) # 解析并返回一个表达式

expression(italic(y) == "37.4" + "1.75" * italic(x) * "," ~ ~italic(r)^2 ~ "=" ~
"0.42")
```

有了字符串表达式之后，就可以将其添加到图形上了。下面设置 x=Inf 和 y=-Inf 将公式放置于图形的右下角，同时对其位置进行上下和左右调整，使其位于绘图区域内（见图 5-27）：

```
sp + annotate("text", label=eqn, parse=TRUE, x=Inf, y=-Inf, hjust=1.1, vjust=-.5)
```

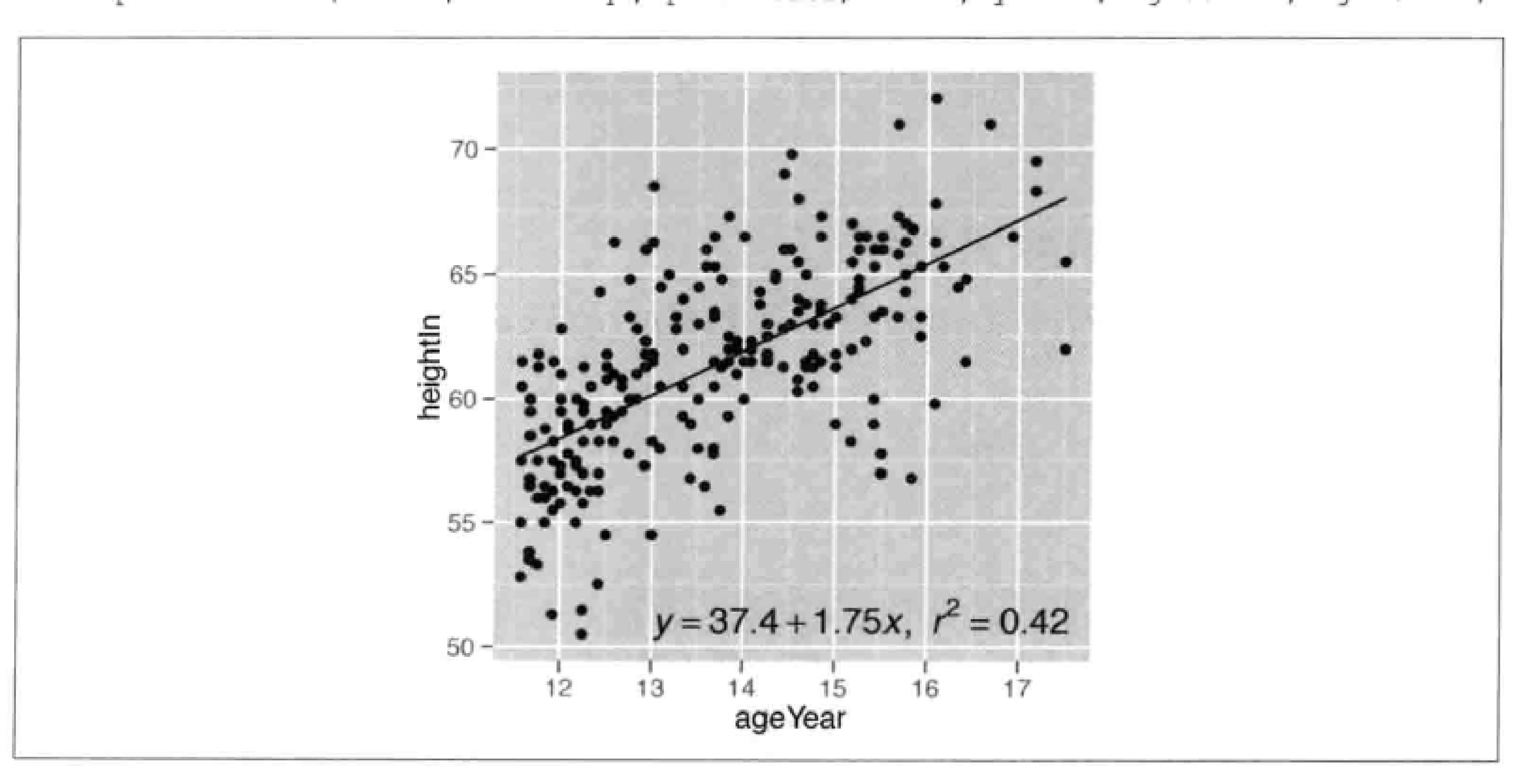


图 5-27　自动生成公式的散点图

另见

R 中数学表达式的语法可能需要花一些时间去学习，更多信息请参见 7.2 节。

5.10 向散点图添加边际地毯

问题

如何向散点图添加边际地毯（Marginal rugs）?

方法

调用 `geom_rug()` 函数即可。下面以 `faithful` 数据集为例（见图 5-28），该数据集包含两列关于“老忠实喷泉”的信息：其中，一列是变量 `eruptions`，记录的是喷泉每次喷发的时长；另一列是 `waiting`，记录的是喷泉在两次喷发之间的时间间隔：

```
ggplot(faithful, aes(x=eruptions, y=waiting)) + geom_point() + geom_rug()
```

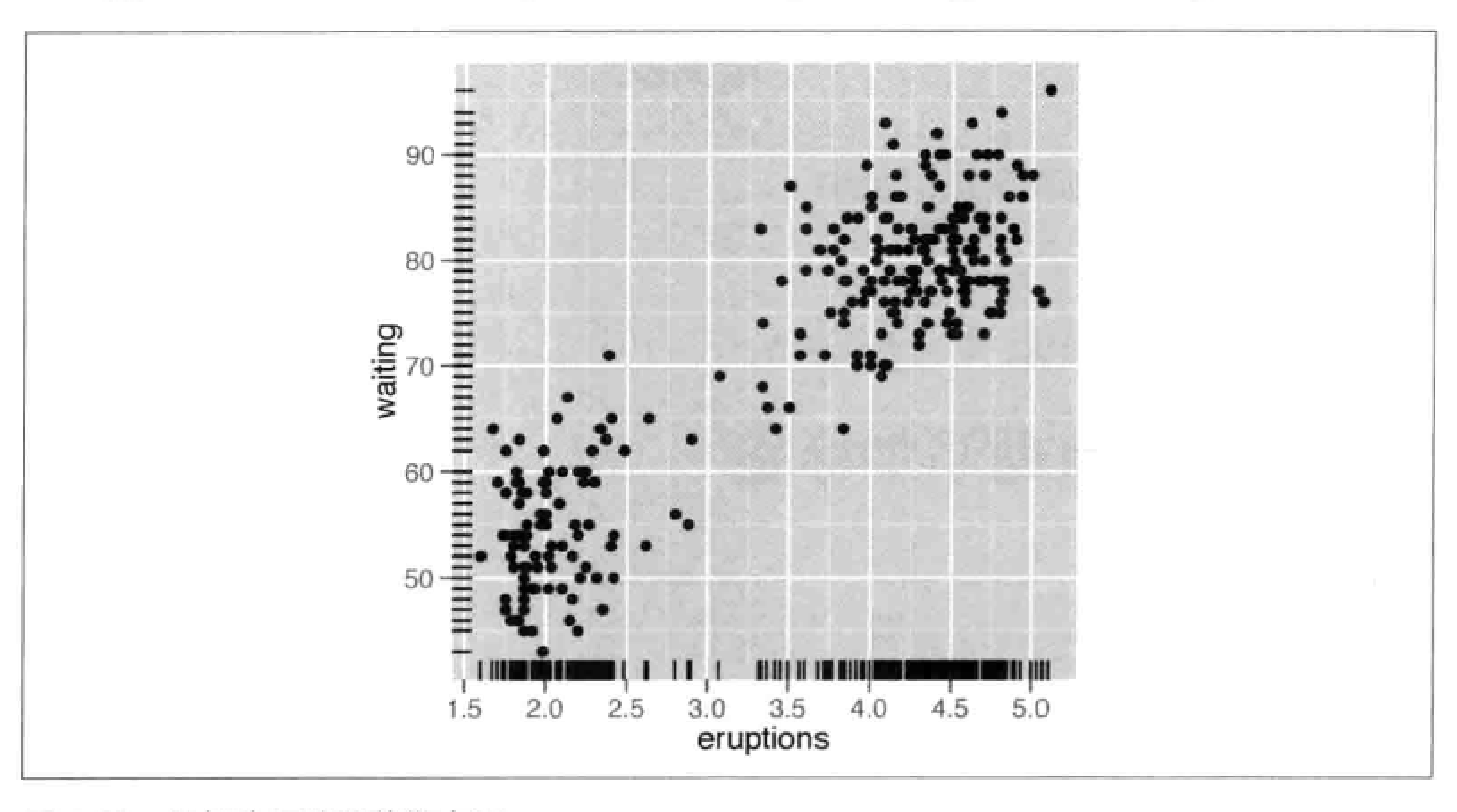


图 5-28 添加边际地毯的散点图

讨论

边际地毯图本质上是一个一维的散点图，它可被用于展示每个坐标轴上数据的分布情况。

对于本例中的数据集，边际地毯传递的信息量十分有限。因为变量 `waiting` 的最小刻度是分钟，因此，图中相应的边际地毯线重叠严重。通过向边际地毯线的位置坐标添加扰动并设定 `size` 减小线宽可以减轻边际地毯线的重叠程度（见图 5-29）。上述操作有助于看清数据的分布情况：

```
ggplot(faithful, aes(x=eruptions, y=waiting)) + geom_point() +
    geom_rug(position="jitter", size=.2)
```

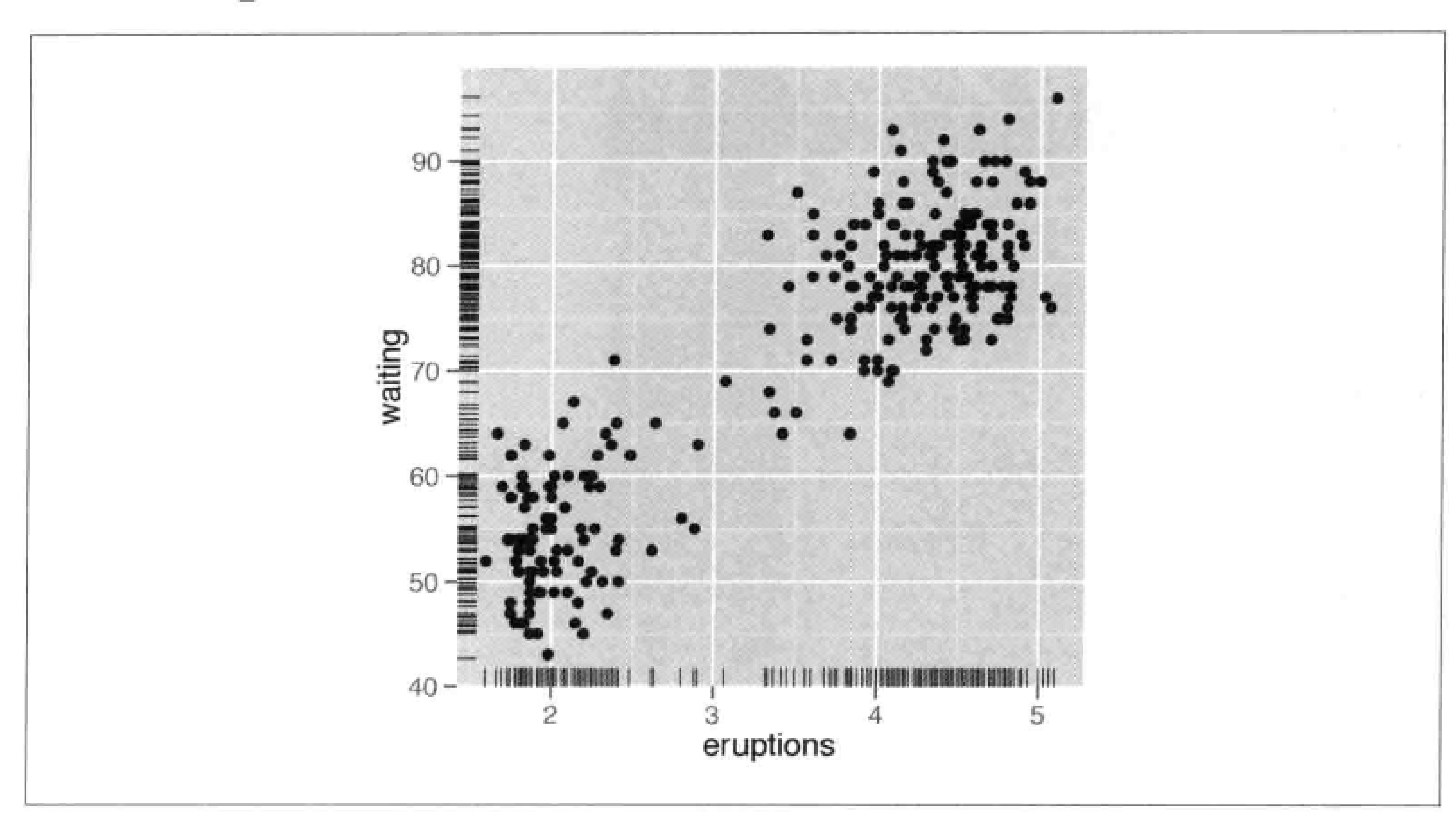


图 5-29　线形更细、添加扰动的边际地毯

另见

更多关于图形重叠的内容，可参见 5.5 节。

5.11　向散点图添加标签

问题

如何向散点图添加标签？

方法

调用 annotate() 函数或者 geom_text() 函数可以为一个或几个数据点添加标签。下面以 countries 数据集为例，对各国医疗保健支出与每千新生儿的婴儿死亡率之间的关系进行可视化。为了便于操作，选取人均支出大于 2000 美元的国家的数据子集进行分析：

```
library(gcookbook) # 为了使用数据
subset(countries, Year==2009 & healthexp>2000)

         Name Code Year      GDP laborrate healthexp infmortality
      Andorra  AND 2009       NA        NA  3089.636          3.1
    Australia  AUS 2009 42130.82      65.2  3867.429          4.2
      Austria  AUT 2009 45555.43      60.4  5037.311          3.6
...
```

```
United Kingdom   GBR 2009  35163.41       62.2  3285.050          4.7
 United States   USA 2009  45744.56       65.0  7410.163          6.6
```

先将基本散点图对象保存在 sp 中，再向其添加其他元素。要手动添加注释，可调用 annotate() 函数，此时，需要指定标签坐标和标签文本（见图 5-30 左图）。可能要尝试多次才能将注释调整到合适的位置。

```
sp <- ggplot(subset(countries, Year==2009 & healthexp>2000),
             aes(x=healthexp, y=infmortality)) +
      geom_point()

sp + annotate("text", x=4350, y=5.4, label="Canada") +
     annotate("text", x=7400, y=6.8, label="USA")
```

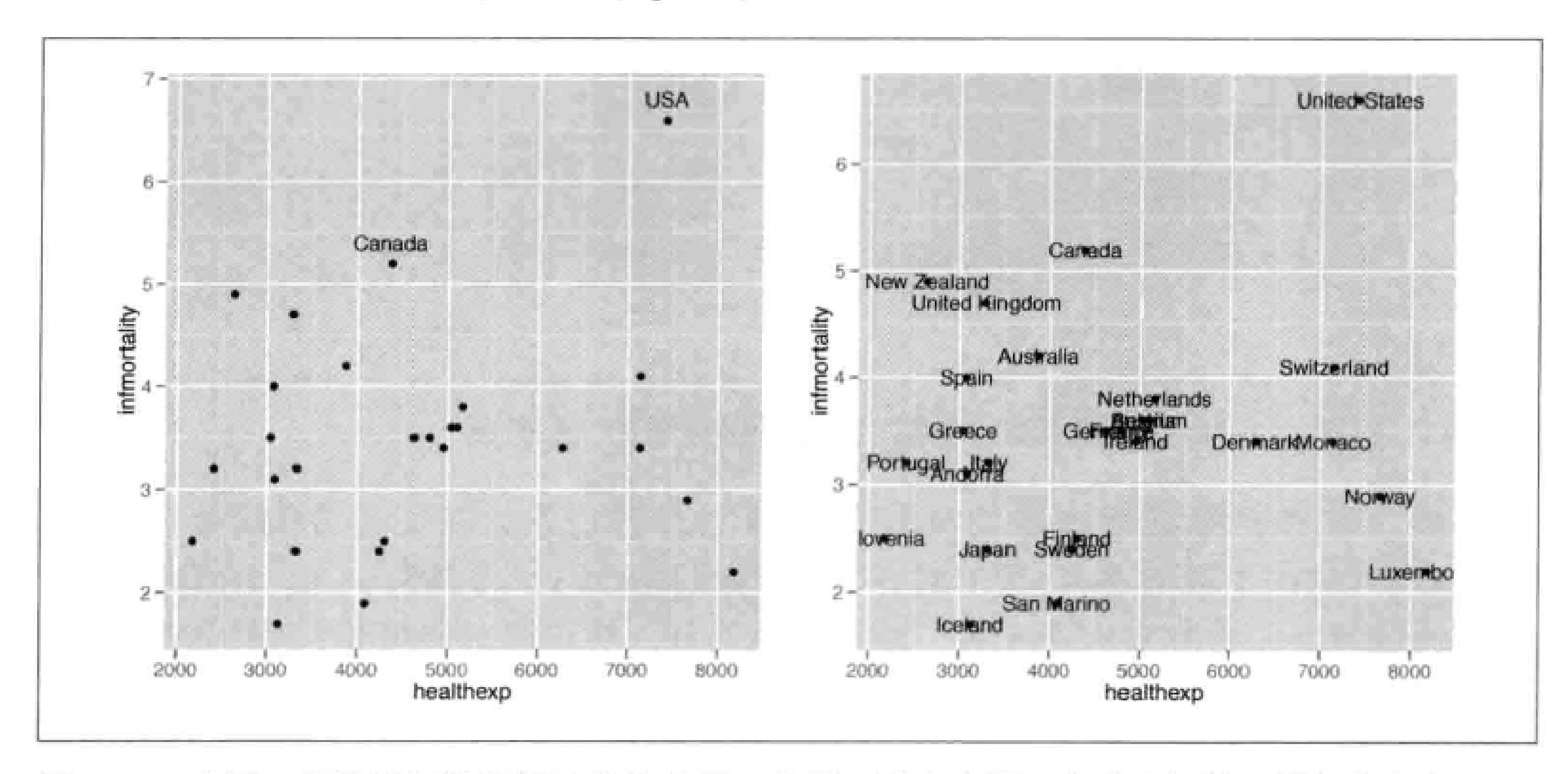


图 5-30　左图：手动设置数据标签的散点图　右图：缩小字号、自动添加数据标签的散点图

要根据数据集自动向散点图添加数据标签（见图 5-30 右图），可以使用 geom_text() 函数，此时，只需映射一个因子型或者字符串型的向量给标签（label）属性。本例中，我们把变量 Name 映射给 label 属性，同时为了避免数据点过于拥挤，我们使用略小一点的字号。默认的标签 size 属性为 5，这个数值并不与字号直接对应。

```
sp + geom_text(aes(label=Name), size=4)
```

讨论

系统自动放置注释时会将其中心置于 *x* 坐标和 *y* 坐标的位置。不过，我们可以对文本位置进行上下或者左右调整，或者两者兼做。

设定 vjust=0 时，标签文本的基线会与数据点对齐（见图 5-31 左图）；设定 vjust=1 时，标签文本的顶部会与数据点对齐。但有时这样还不够，我们还可以通过增加或者减少 vjust 参数的值来调高或者调低文本标签的位置；也可以通过对 y 的映射增加或

减小一个值得到相同的效果（见图 5-31 右图）。

```
sp + geom_text(aes(label=Name), size=4, vjust=0)

# 增加一些 y 的取值
sp + geom_text(aes(y=infmortality+.1, label=Name), size=4, vjust=0)
```

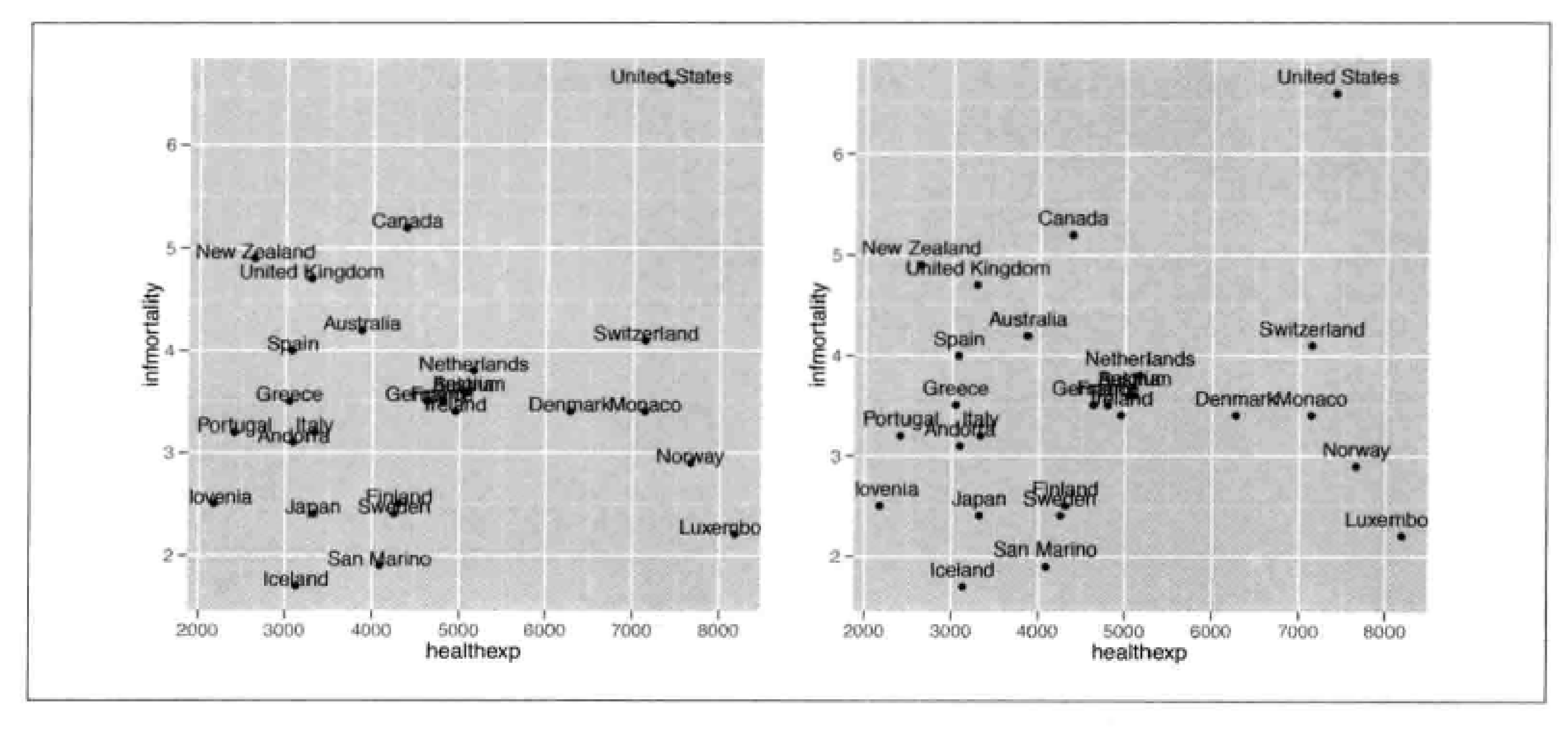


图 5-31　左图：vjust=0 的散点图　右图：向 *y* 轴坐标增加一些数值

有时候，有必要根据数据点的位置令注释左对齐或右对齐。要左对齐，可设置 hjust=0（见图 5-32 左图）；要右对齐，可设定 hjust=1。与调整 vjust 的情形类似，图中的标签会与数据点有所重叠。然而，这时候，我们最好不要通过增加或者减少 hjust 的值对此进行修正，因为调整 hjust 的值时，系统会按照文本标签长度的一定比例来移动文本标签的位置，这时候，较长的文本标签会比短文本标签移动的位置更大。此时，最好将 hjust 设定为 0 或者 1，然后，通过对 x 增加或者减去一个值来调整文本标签的位置（见图 5-32 右图）：

```
sp + geom_text(aes(label=Name), size=4, hjust=0)

sp + geom_text(aes(x=healthexp+100, label=Name), size=4, hjust=0)
```



如果绘图时用的是对数坐标轴，要想将文本标签移动同样的位置，就不能通过增加 x 或 y 的数值来实现了。此时需要令 x 或者 y 乘以一个数值才行。

如果只想给为数不多的几个点添加标签，但希望 R 自动设定标签位置的话，可以给数据框增加一个只包含拟使用的标签的新列。一个可行方案是：首先，将所用数据复制一个副本，并将列 Name 复制为 Name1：

```
cdat <- subset(countries, Year==2009 & healthexp>2000)

cdat$Name1 <- cdat$Name
```

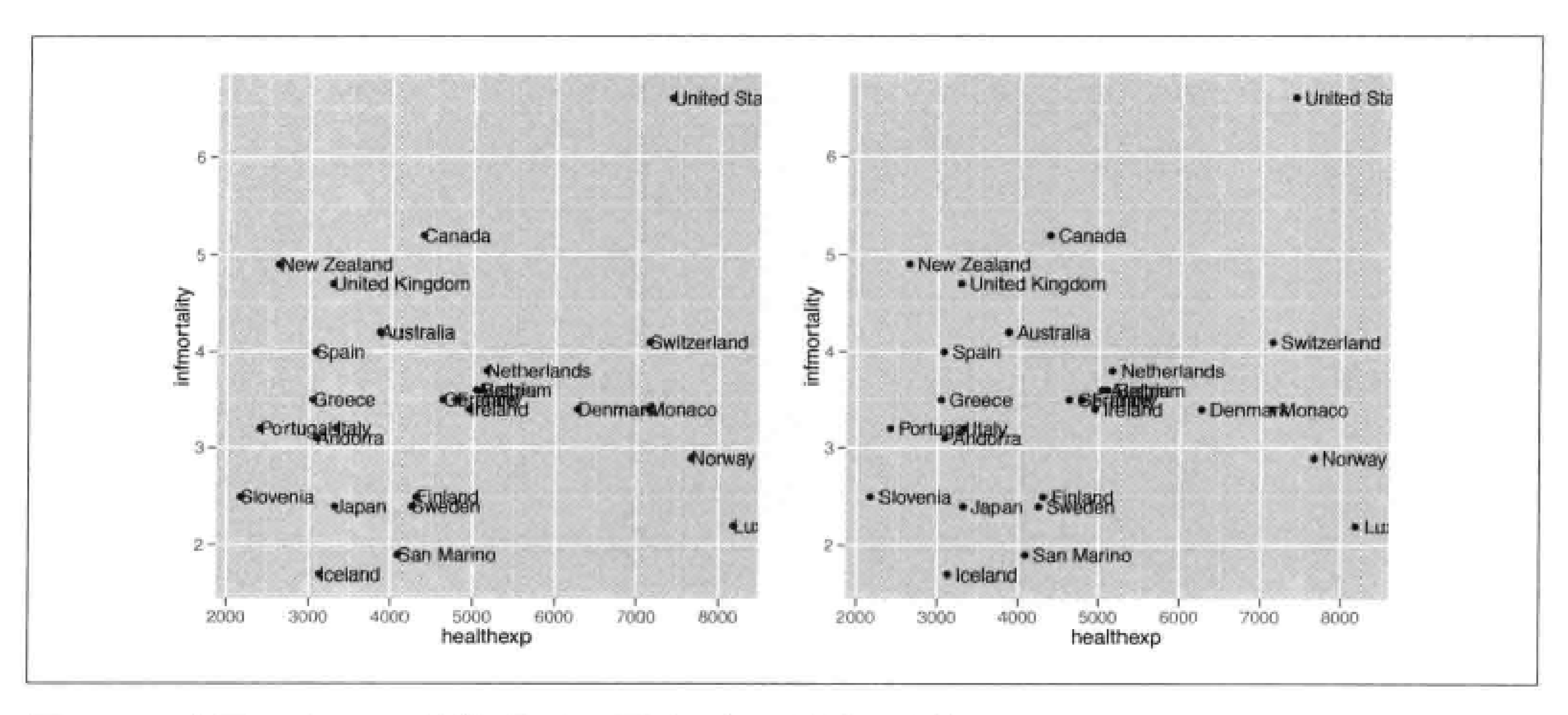


图 5-32　左图：`hjust=0` 的散点图　右图：向 x 添加一个值

接下来，用 `%in%` 运算符找出绘图时希望保有的标签所处的位置。本操作将返回一个逻辑向量，该向量标识了 `cdat$Name1` 中哪些元素出现在第二个向量中，其中第二个向量指定的是我们希望标示出来的国家的名字：

```
idx <- cdat$Name1 %in% c("Canada", "Ireland", "United Kingdom", "United States",
                         "New Zealand", "Iceland", "Japan", "Luxembourg",
                         "Netherlands", "Switzerland")
idx

 [1] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE
[13] FALSE  TRUE  TRUE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
[25]  TRUE  TRUE  TRUE
```

根据上面的逻辑向量用 `NA` 重写变量 `Name1` 中的其他取值：

```
cdat$Name1[!idx] <- NA
```

得到的结果如下：

```
cdat

           Name Code Year      GDP laborrate healthexp infmortality          Name1
        Andorra  AND 2009       NA        NA  3089.636          3.1           <NA>
      Australia  AUS 2009 42130.82      65.2  3867.429          4.2           <NA>
...
    Switzerland  CHE 2009 63524.65      66.9  7140.729          4.1    Switzerland
 United Kingdom  GBR 2009 35163.41      62.2  3285.050          4.7 United Kingdom
  United States  USA 2009 45744.56      65.0  7410.163          6.6  United States
```

现在，可以绘制图形了（见图 5-33）。这次，增大 x 轴的范围使得文本标签可以漂亮地展示出来：

```
ggplot(cdat, aes(x=healthexp, y=infmortality)) +
    geom_point() +
    geom_text(aes(x=healthexp+100, label=Name1), size=4, hjust=0) +
    xlim(2000, 10000)
```

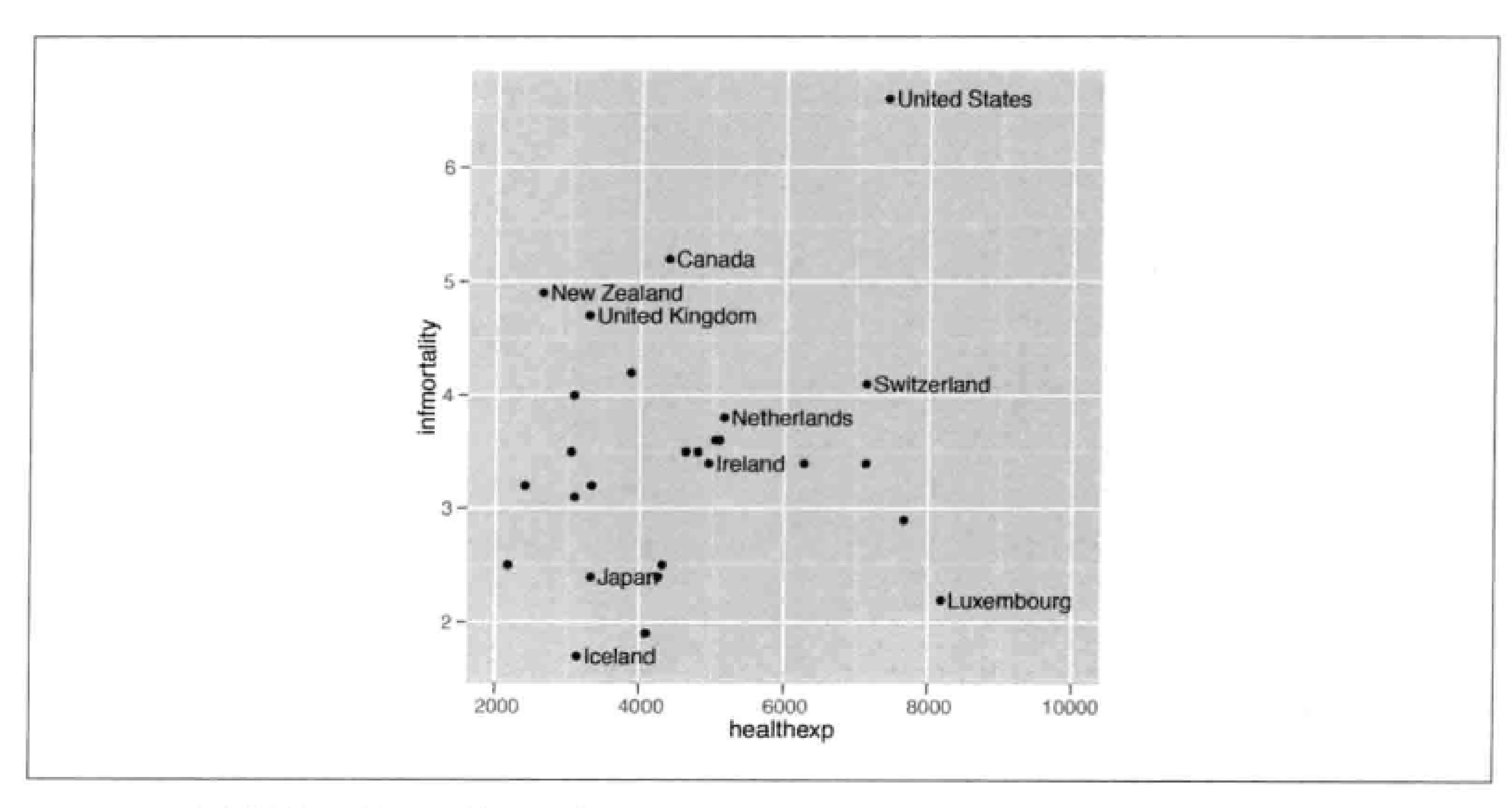


图 5-33 选定标签、扩展 x 轴的散点图

如果一些个别的位置需要调整，我们有两个选择：第一个选择是复制 x 轴坐标和 y 轴坐标对应的列，修改它们的数值并以此调整文本的位置。当然，绘制数据点所用的坐标必须是原始的数值！另一个选择是把图形输出为矢量格式，比如 PDF 或者 SVG（参见 14.1 节和 14.2 节的内容），再用 Illustrator 或者 Inkscape 软件对其进行编辑。

另见

更多关于控制文本样式的内容，可参见 9.2 节。 如果想手动编辑 PDF 或者 SVG 文件，可参见 14.4 节的内容。

5.12 绘制气泡图

问题

如何绘制气泡图，并令点的面积正比于变量值？

方法

调用 `geom_point()` 和 `scale_size_area()` 函数即可绘制气泡图。下面以 `countries` 数据集的子集为例：

```
library(gcookbook) # 为了使用数据

cdat <- subset(countries, Year==2009 &
   Name %in% c("Canada", "Ireland", "United Kingdom", "United States",
               "New Zealand", "Iceland", "Japan", "Luxembourg",
               "Netherlands", "Switzerland"))
cdat
```

```
                 Name Code Year       GDP laborrate healthexp infmortality
1733           Canada  CAN 2009  39599.04      67.8  4379.761          5.2
4436          Iceland  ISL 2009  37972.24      77.5  3130.391          1.7
4691          Ireland  IRL 2009  49737.93      63.6  4951.845          3.4
4946            Japan  JPN 2009  39456.44      59.5  3321.466          2.4
5864       Luxembourg  LUX 2009 106252.24      55.5  8182.855          2.2
7088      Netherlands  NLD 2009  48068.35      66.1  5163.740          3.8
7190      New Zealand  NZL 2009  29352.45      68.6  2633.625          4.9
9587      Switzerland  CHE 2009  63524.65      66.9  7140.729          4.1
10454  United Kingdom  GBR 2009  35163.41      62.2  3285.050          4.7
10505   United States  USA 2009  45744.56      65.0  7410.163          6.6
```

如果只是将变量 GDP 映射给 size 属性，则 GDP 的值被映射给了点的半径（见图 5-34 左图），这并不是我们想要的结果；此外，当变量值增加一倍时，对应的点的面积会变为原来的四倍，因此，这种结果会对人们理解数据产生误导。我们更想将 GDP 映射给点的面积，可以调用 scale_size_area() 来完成这一操作（见图 5-34 右图）：

```
p <- ggplot(cdat, aes(x=healthexp, y=infmortality, size=GDP)) +
    geom_point(shape=21, colour="black", fill="cornsilk")

# 将 GDP 映射给半径 (scale_size_continuous 的默认值 )
p

# 将 GDP 映射给面积，得到一个略大的圆圈
p + scale_size_area(max_size=15)
```

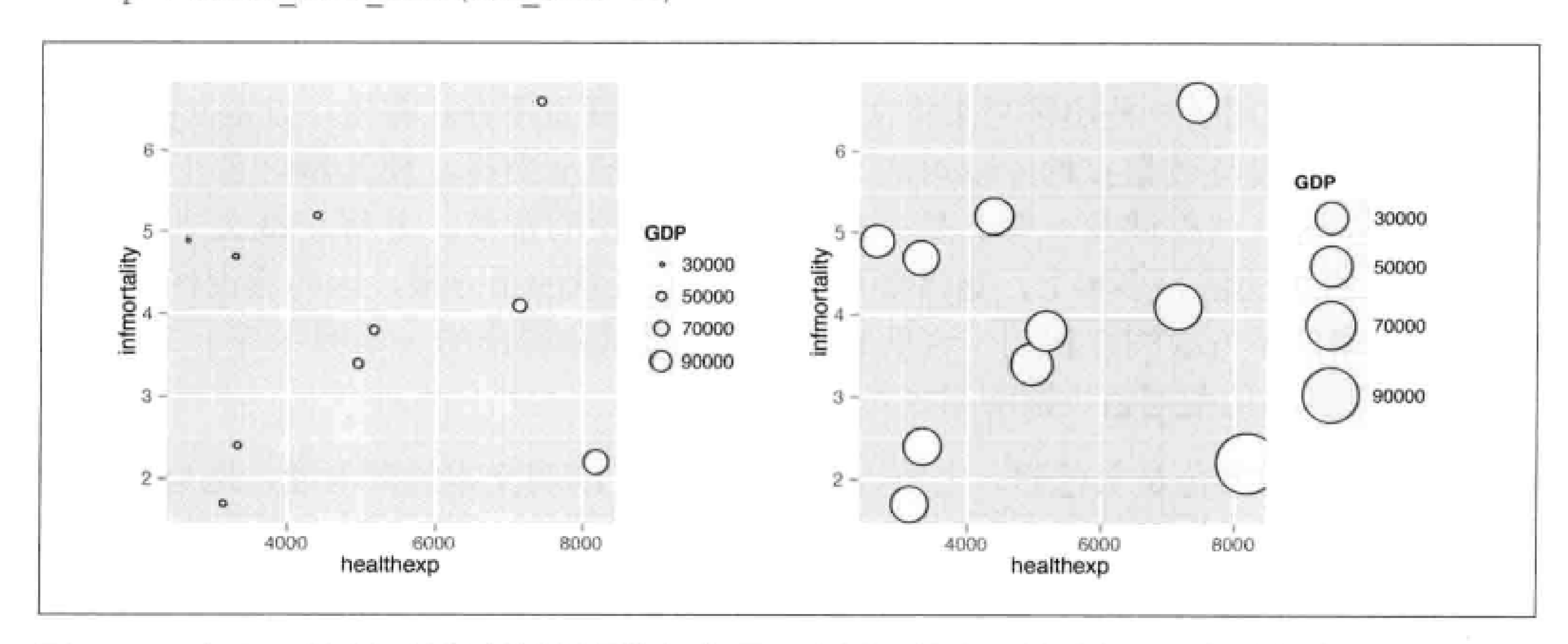


图 5-34　左图：把变量值映射给半径的气泡图　右图：把变量值映射给面积的气泡图

讨论

本例中的气泡图实际上还是散点图，但气泡图也有其他用法。比如，当 x 轴和 y 轴皆为分类变量时，气泡图可以用来表示网格点上的变量值，如图 5-35 所示：

```
# 对男性组和女性组求和
hec <- HairEyeColor[,,"Male"] + HairEyeColor[,,"Female"]
```

```
# 转化为长格式（long format）
library(reshape2)
hec <- melt(hec, value.name="count")

ggplot(hec, aes(x=Eye, y=Hair)) +
    geom_point(aes(size=count), shape=21, colour="black", fill="cornsilk") +
    scale_size_area(max_size=20, guide=FALSE) +
    geom_text(aes(y=as.numeric(Hair)-sqrt(count)/22, label=count), vjust=1,
              colour="grey60", size=4)
```

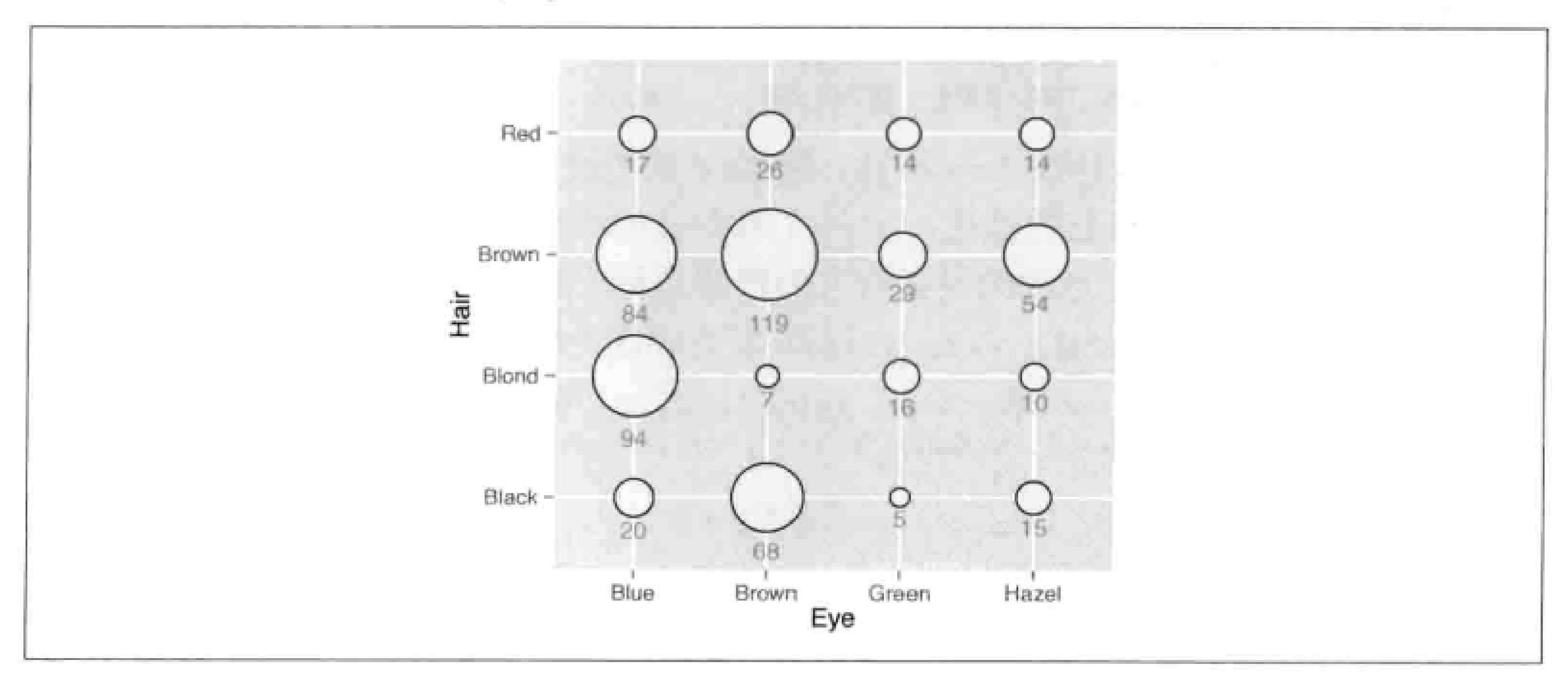


图 5-35 坐标轴对应于分类变量的气泡图及文本标签

本例中，我们使用了一些小技巧来将文本标签置于圆圈正下方。首先，设定 vjust=1，将文本标签顶端与数据点的 y 轴对应。接下来，调整 y 坐标，使其刚好位于每个圆圈的底部。这一过程涉及一些计算：提取 Hair 变量的数值，并将其减去一个与变量 count 有关的值。事实上，这需要计算 count 变量的平方根，因为圆圈的半径跟 count 变量的平方根线性相关。这里的除数是通过试错得出的（本例中等于 22）；这个值取决于具体的数据值、圆圈半径及文本大小。

圆圈下面的文本颜色是灰色的。之所以这么做是为了削弱文本的突出性，避免影响圆圈的展示，但同时如果想知道圆圈对应的具体数值，文本也能够在图中观察得出。

另见

为圆圈添加标签的内容可参见 5.11 节和 7.1 节。 在散点图中将变量映射给其他图形属性的方法可参见 5.4 节。

5.13 绘制散点图矩阵

问题

如何绘制散点图矩阵？

方法

散点图矩阵是一种对多个变量两两之间关系进行可视化的有效方法。调用 R 基础绘图系统中的 pairs() 函数可以绘制散点图矩阵。

这里以 countries 数据集的子集为例。我们从 countries 数据集中选取出 2009 年的数据且只保留几个相关列：

```
library(gcookbook) # 为了使用数据
c2009 <- subset(countries, Year==2009,
                select=c(Name, GDP, laborrate, healthexp, infmortality))

c2009
```

```
       Name        GDP laborrate healthexp infmortality
Afghanistan         NA      59.8  50.88597        103.2
    Albania  3772.6047      59.5 264.60406         17.2
    Algeria  4022.1989      58.5 267.94653         32.0
...
     Zambia  1006.3882      69.2  47.05637         71.5
   Zimbabwe   467.8534      66.8        NA         52.2
```

我们用第 2-5 列数据绘制散点图矩阵（见图 5-36）——对变量 Name 绘图的意义不大，而且会出现奇怪的结果：

```
pairs(c2009[,2:5])
```

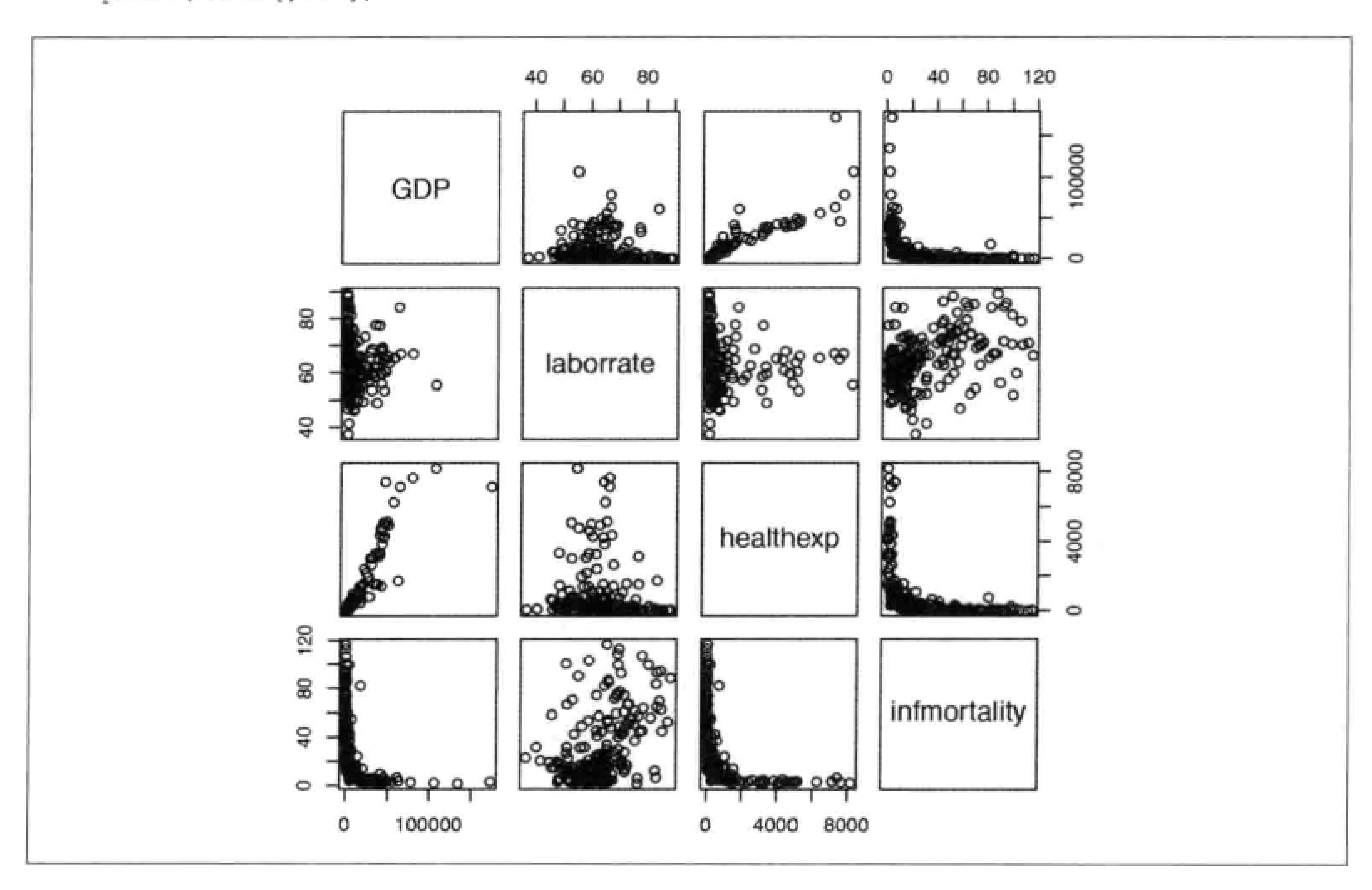


图 5-36　散点图矩阵

讨论

此处，我们没有使用 ggplot2 是因为它不能绘制散点图矩阵（至少绘制的效果不佳）。

上述绘图过程中也可以使用自定义的面板函数。我们定义一个 panel.cor 函数来展示变量两两之间的相关系数以代替默认的散点图。相关系数较大的位置将对应于较大的字体。现在暂时不需要关心函数的细节——先把代码粘贴到 R 会话或者脚本中：

```
panel.cor <- function(x, y, digits=2, prefix="", cex.cor, ...) {
    usr <- par("usr")
    on.exit(par(usr))
    par(usr = c(0, 1, 0, 1))
    r <- abs(cor(x, y, use="complete.obs"))
    txt <- format(c(r, 0.123456789), digits=digits)[1]
    txt <- paste(prefix, txt, sep="")
    if(missing(cex.cor)) cex.cor <- 0.8/strwidth(txt)
    text(0.5, 0.5, txt, cex =  cex.cor * (1 + r) / 2)
}
```

为了在面板的对角线上展示各个变量的直方图，我们定义 panel.hist 函数：

```
panel.hist <- function(x, ...) {
    usr <- par("usr")
    on.exit(par(usr))
    par(usr = c(usr[1:2], 0, 1.5) )
    h <- hist(x, plot = FALSE)
    breaks <- h$breaks
    nB <- length(breaks)
    y <- h$counts
    y <- y/max(y)
    rect(breaks[-nB], 0, breaks[-1], y, col="white", ...)
}
```

上面的函数都取自于 pairs 函数的帮助页面，为方便起见，也可以打开 pairs 函数的帮助页面，复制、粘贴相关代码。不过，我们对这一版本的 panel.cor 函数的最后一行进行了微小的修正，以使得散点矩阵图中的字体差异不像原始代码那么大。

定义了这些函数之后，我们可以调用它们来绘制散点图矩阵。令 pairs() 函数在面板的上三角执行 panel.cor 函数；在面板的对角线执行 panel.hist 函数。

绘图时也额外增加了一点东西：在面板的下三角执行 panel.smooth 函数。该函数将在散点图矩阵的下三角绘制散点图，并添加一个 LOWESS 平滑曲线，如图 5-37 所示（5.6 节中介绍了 LOESS，此处的 LOWESS 与 LOESS 略有不同，不过，对于这种初步的探索可视化方法而言，两者的区别并不重要）：

```
pairs(c2009[,2:5], upper.panel = panel.cor,
                   diag.panel  = panel.hist,
                   lower.panel = panel.smooth)
```

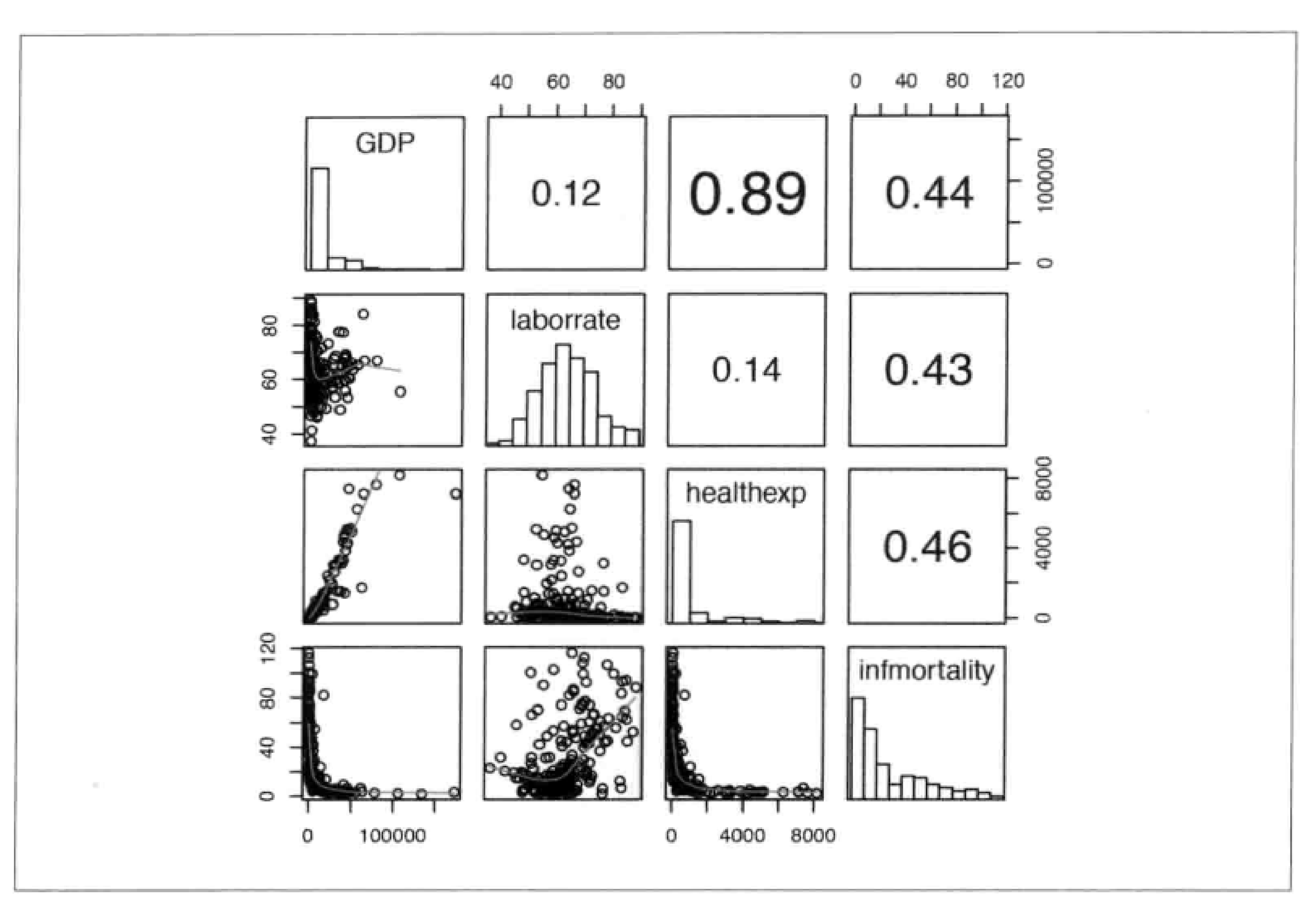


图 5-37 上三角是相关系数，下三角是平滑曲线，对角线是直方图的散点图矩阵

也许，我们会希望用线性模型代替 LOWESS 模型，panel.lm 函数可以完成该操作（与前面的面板函数不同，这里的函数不在 pairs 函数的帮助页面中）：

```
panel.lm <- function (x, y, col = par("col"), bg = NA, pch = par("pch"),
                      cex = 1, col.smooth = "black", ...) {
    points(x, y, pch = pch, col = col, bg = bg, cex = cex)
    abline(stats::lm(y ~ x),  col = col.smooth, ...)
}
```

这次，系统默认的线条颜色不再是红色，而是黑色。调用函数 pairs() 时（与函数 panel.smooth 配合使用）设定 col.smooth 参数可以对线条颜色进行修改。

为了便于辨认数据点，我们在图中使用更小一些的点（见图 5-38）。该操作可以通过设定 pch="." 来完成：

```
pairs(c2009[,2:5], pch=".",
                   upper.panel = panel.cor,
                   diag.panel  = panel.hist,
                   lower.panel = panel.lm)
```

cex 参数可以控制图中点的大小。cex 参数的默认值是 1，其值越大，相应的数据点也越大，反之亦然。如果参数 cex 小于 0.5，图形输出为 PDF 文件时可能无法很好地渲染。

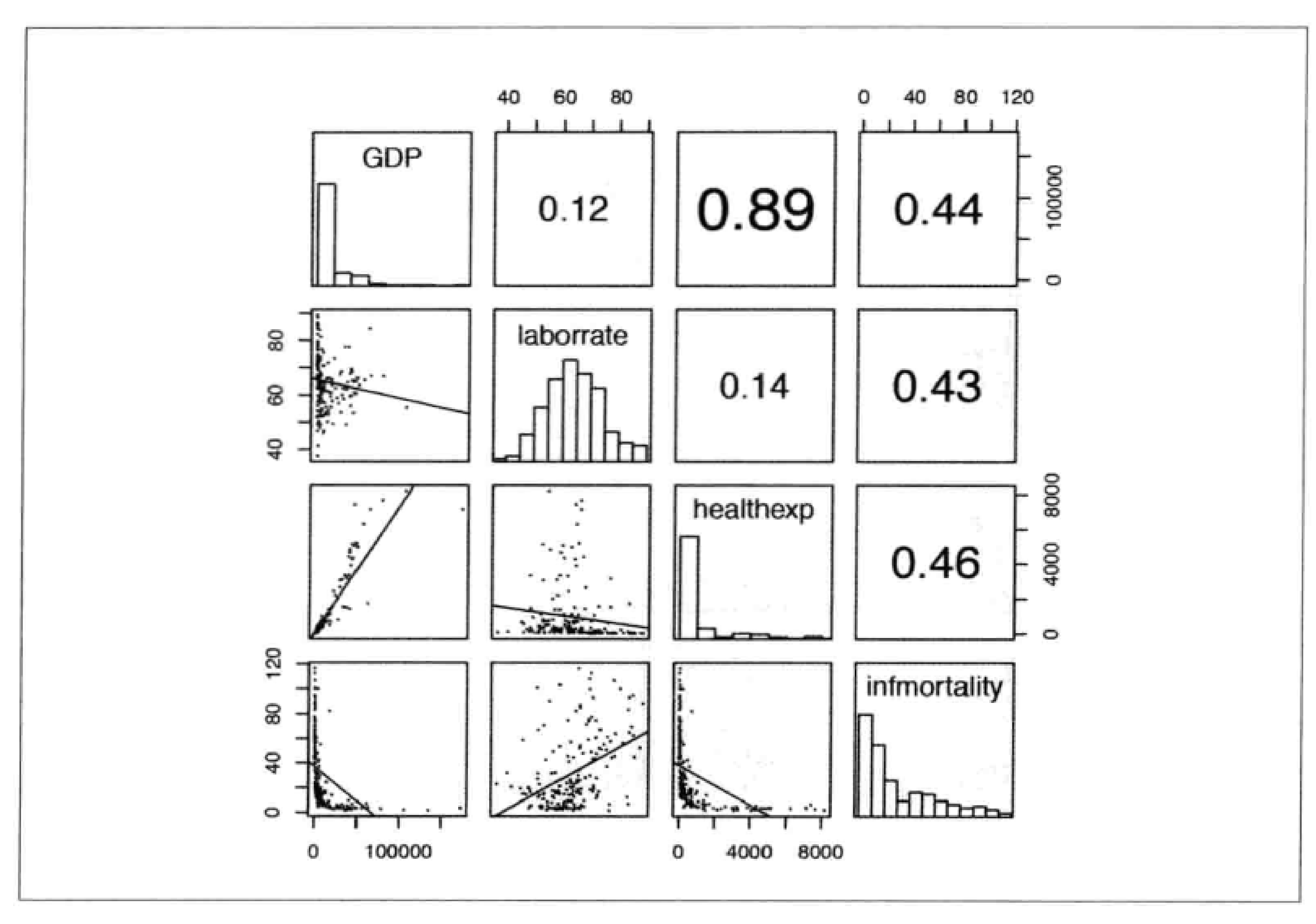


图 5-38　使用更小的数据点及线性拟合线的散点图矩阵

更多

关于创建相关系数矩阵的内容，可参见 13.1 节。

GGally 包中的 `ggpairs()` 函数也可以绘制散点图矩阵。

第6章

描述数据分布

本章将探寻一些对数据分布进行可视化的方法。

6.1 绘制简单直方图

问题

如何绘制直方图？

方法

运行 geom_histogram() 函数并映射一个连续型变量到参数 x（见图 6-1）：

```
ggplot(faithful, aes(x=waiting)) + geom_histogram()
```

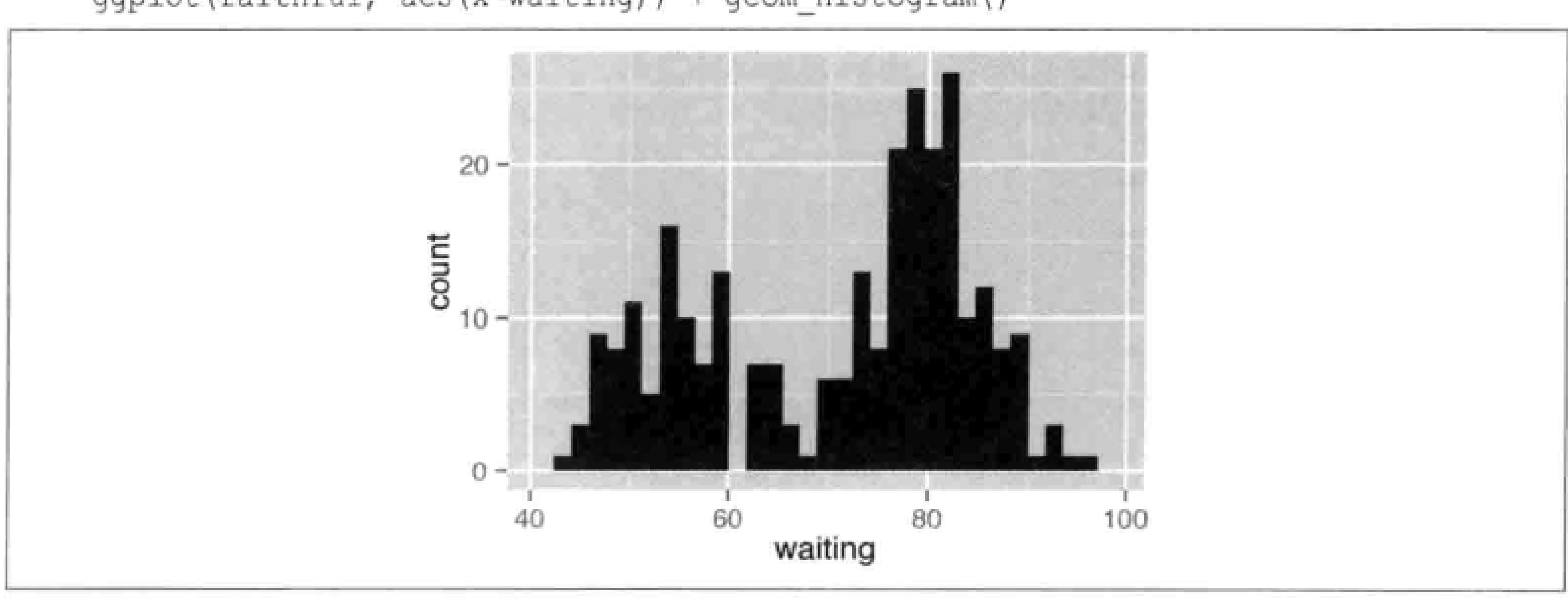


图 6-1 简单直方图

讨论

geom_histogram() 函数只需要数据框的其中一列或者一个单独的数据向量作为参

数。以 faithful 数据集为例，该数据集包含了两列描述老忠实喷泉的信息：第一列 eruptions，描述老忠实喷泉每次喷发的时长；第二列 waiting，描述两次喷发之间的间隔。下面仅以列 waiting 为例：

```
faithful

eruptions waiting
    3.600      79
    1.800      54
    3.333      74
...
```

如果想快速地看一下未包含在数据框中的数据的直方图，可以在运行上述命令时，将数据框参数设定为 NULL，同时，向 ggplot() 函数传递一个向量作为参数。下面的代码与之前的运行结果相同：

```
# 将变量值保存为一个基本向量
w <-faithful$waiting

ggplot(NULL, aes(x=w)) + geom_histogram()
```

默认情况下，数据将被切分为 30 组，这样分组可能太过精细，也可能太过粗糙，这取决于数据的实际情况。我们可以通过组距（binwidth）参数来调整数据的分组数目，或者将数据切分为指定的分组数目。直方图默认的填充色是黑色且没有边框线，这使得我们难以看清各个条形对应的变量值，因此，可以调整一下直方图的颜色设置，如图 6-2 所示：

```
# 设定组距为 5
ggplot(faithful, aes(x=waiting)) +
    geom_histogram(binwidth=5, fill="white", colour="black")

# 将 x 的取值切分为 15 组
binsize <- diff(range(faithful$waiting))/15
ggplot(faithful, aes(x=waiting)) +
    geom_histogram(binwidth=binsize, fill="white", colour="black")
```

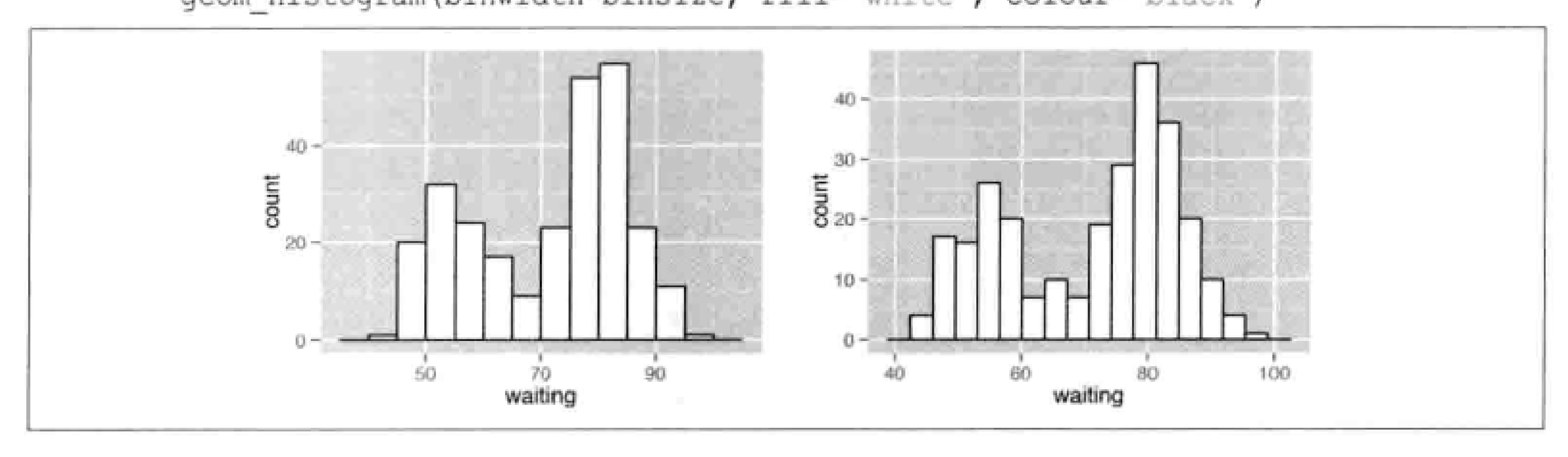


图 6-2　左图：组距为 5，且修改填充色的直方图　右图：分组数为 15 的直方图

有时，直方图的外观会非常依赖于组距及组边界。图 6-3 中，我们将组距设定为 8。同时设定分组原点（origin）参数令左图的组边界分别位于 31、39、47 等；右图中，对 origin 参数增加 4，令组边界分别位于 35、43、51 等：

```
h <- ggplot(faithful, aes(x=waiting)) # 将基本绘图结果存为变量以便于重复使用
```



```
h + geom_histogram(binwidth=8, fill="white", colour="black", origin=31)
h + geom_histogram(binwidth=8, fill="white", colour="black", origin=35)
```

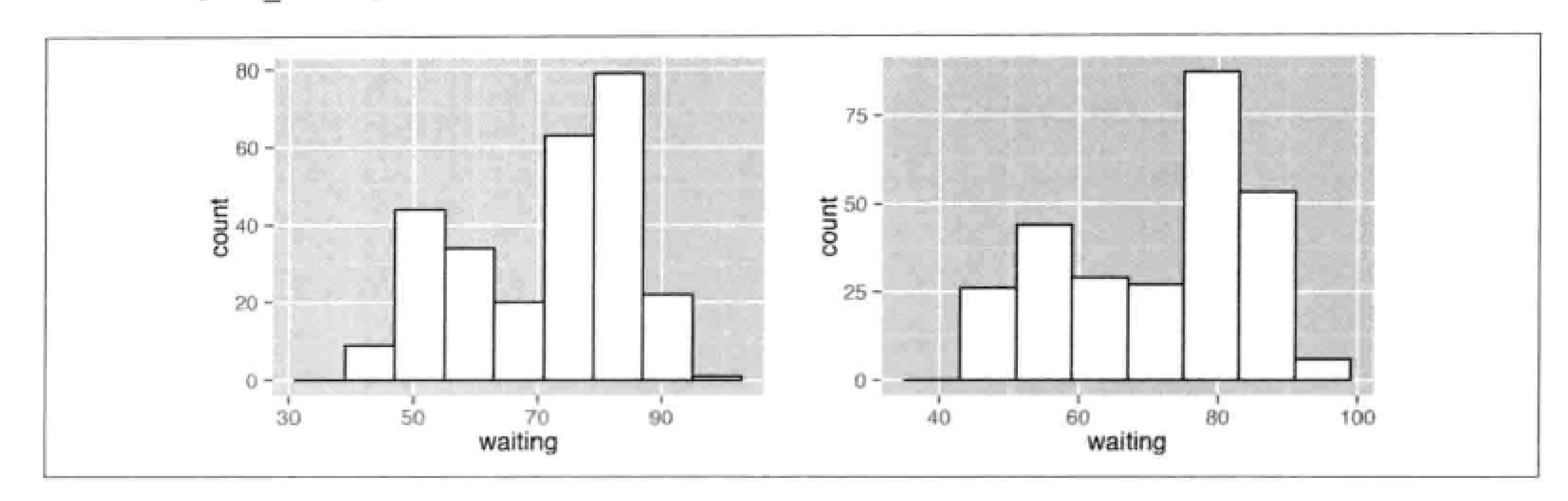


图 6-3　分组原点（origin）对应于 31 和 35 时的直方图

两图对应的分组数目相同，但绘图结果差异很大。本例中的的 `faithful` 数据集共包含 272 个观测值，数据量并不算少；当数据量较小时，对分组边界的影响将会更大。因此，绘制图形时，最好尝试一下不同的分组数目和分组边界。

当数据集包含离散型数据时，直方图的非对称性不可忽视。数据分组时，各分组区间左闭右开。比如，当分组边界为 1、2、3 等时，对应的分组区间为 [1,2)、[2,3)、[3,4) 等。换言之，第一个分组区间包含 1，但不包含 2，第二个分组区间包含 2，但不包含 3。

运行代码 `geom_bar(stat="bin")` 也能得到相同的结果，不过，`geom_histogram()` 函数的操作过程更易于解释。

另见

绘制多个数据对应的分布时，频数多边形（frequency polygon）是一个更佳的方案，因为，它避免了各个条形之间相互干扰。相关内容参见 6.5 节。

6.2　基于分组数据绘制分组直方图

问题

如何绘制多组数据的直方图？

方法

运行 `geom_histogram()` 函数并使用分面绘图即可，如图 6-4 所示：

```
library(MASS) # 为了使用数据

# 使用 smoke 作为分面变量
ggplot(birthwt, aes(x=bwt)) + geom_histogram(fill="white",colour="black")+
    facet_grid(smoke ~ .)
```

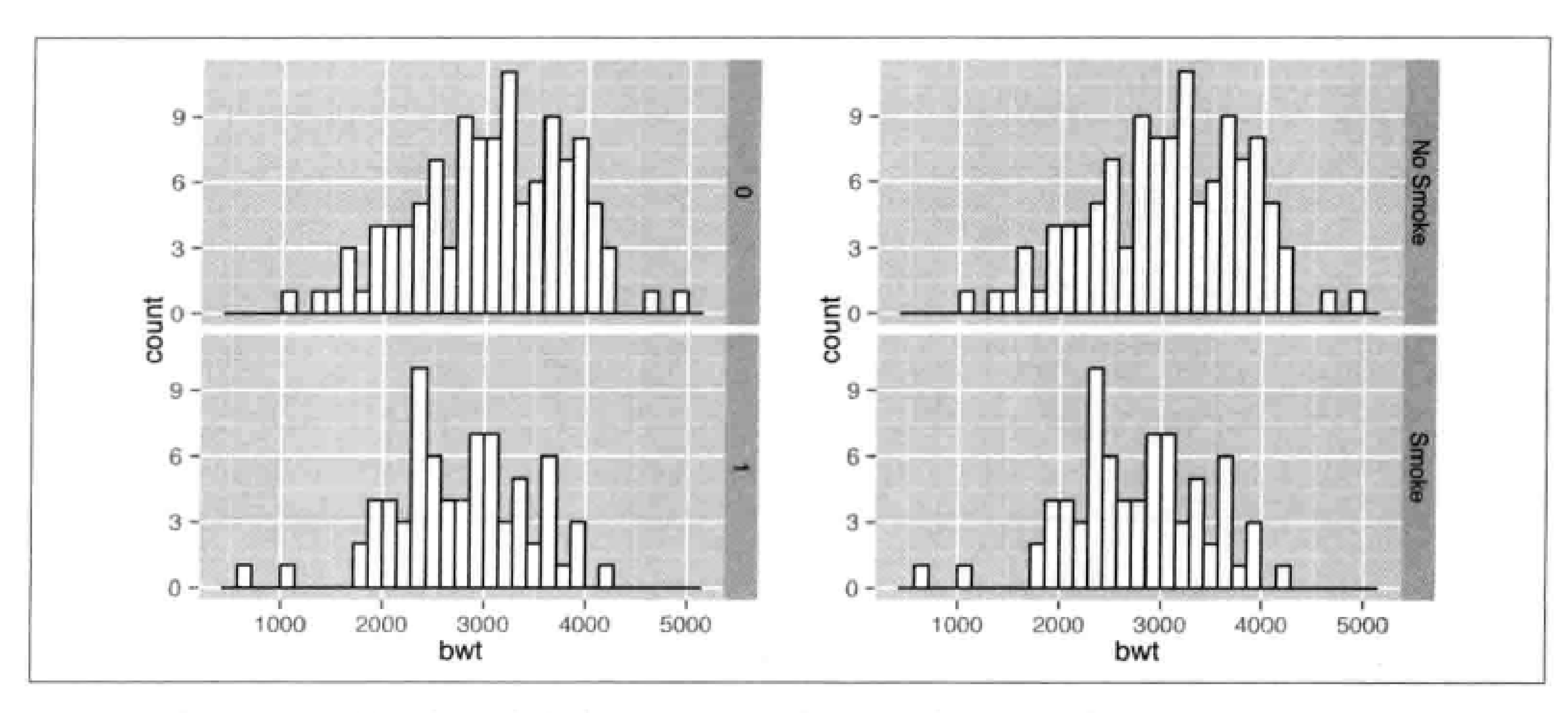


图 6-4　左图：分面数目为 2 的直方图　右图：修改分面标签的直方图

讨论

绘制上图时，要求所有用到的数据都包含在一个数据框里，且数据框其中一列是可用于分组的分类变量。

这里以 `birthwt` 数据集为例。该数据集包含的是关于婴儿出生体重及一系列导致出生体重过低的危险因子的数据：

```
birthwt

low age lwt race smoke ptl ht ui ftv  bwt
  0  19 182    2     0   0  0  1   0 2523
  0  33 155    3     0   0  0  0   3 2551
  0  20 105    1     1   0  0  0   1 2557
...
```

分面绘图有一个问题，即分面标签只有 0 和 1，且没有指明这个标签是变量 `smoke` 的取值。想要修改标签，我们需要修改因子水平的名称。首先列出现有的因子水平，然后，依照相同的顺序向它们赋予新的名字：

```
birthwt1 <- birthwt # 复制一个数据副本

# 将 smoke 转化为因子
birthwt1$smoke <- factor(birthwt1$smoke)
levels(birthwt1$smoke)

"0""1"

library(plyr) # 为了使用 revalue() 函数
birthwt1$smoke <- revalue(birthwt1$smoke, c("0"="No Smoke", "1"="Smoke"))
```

重新绘图，图形中为新的分面标签（见图 6-4 右图）。

```
ggplot(birthwt1, aes(x=bwt)) + geom_histogram(fill="white", colour="black") +
```

```
    facet_grid(smoke ~ .)
```

分面绘图时，各分面对应的 y 轴标度是相同的。当各组数据包含的样本数目不同时，可能会难以比较各组数据的分布形状。我们可以看看按照 race 对出生体重进行分组并分面绘图的结果（见图 6-5 左图）：

```
ggplot(birthwt, aes(x=bwt)) + geom_histogram(fill="white", colour="black") +
    facet_grid(race ~ .)
```

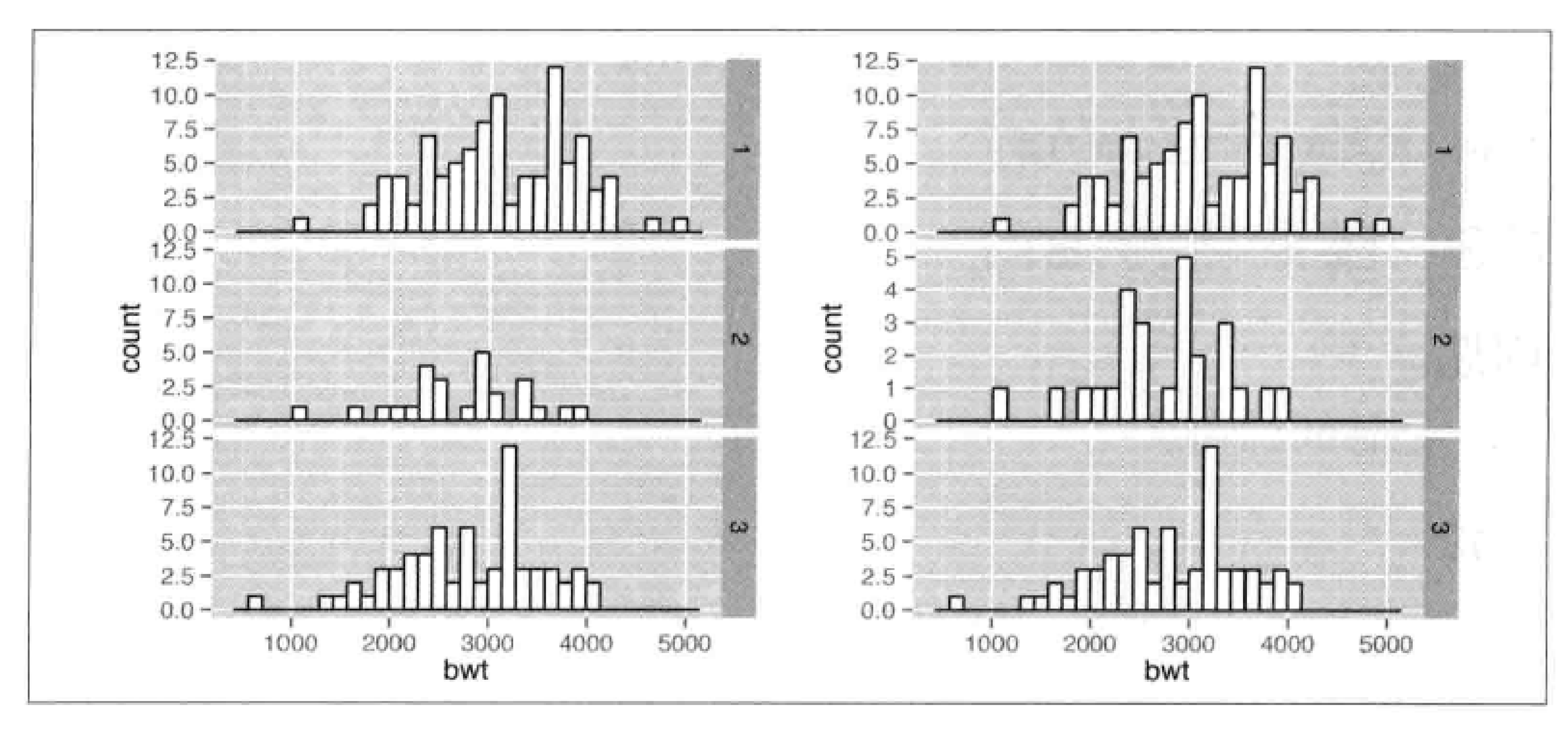


图 6-5　左图：坐标轴对应于默认固定标度的直方图　右图：令 scales="free" 的直方图

设置参数 scales="free" 可以单独设定各个分面的 y 轴标度。注意：这种设置只适用于 y 轴标度，x 轴的标度仍是固定的，因为各个分面的直方图是依照 x 轴进行对齐的。

```
ggplot(birthwt, aes(x=bwt)) + geom_histogram(fill="white", colour="black") +
    facet_grid(race ~ ., scales="free")
```

分组绘图的另一种做法是把分组变量映射给 fill，如图 6-6 所示。此处的分组变量必须是因子型或者字符型的向量。对于 birthwt 数据集，变量 smoke 是合适的分组变量，由于其被存储为数值型，所以，我们使用前面创建的 birthwt1 数据集，该数据集中的 smoke 变量是因子型变量：

```
# 把 smoke 转化为因子
birthwt1$smoke <- factor(birthwt1$smoke)

# 把 smoke 映射给 fill，取消条形堆叠，并使图形半透明

ggplot(birthwt1, aes(x=bwt, fill=smoke)) +
    geom_histogram(position="identity", alpha=0.4 )
```

语句 position="identity" 很重要。没有它，ggplot() 函数会将直方图的条形进行垂直堆积，这样的话更难以看清每组数据的分布信息。

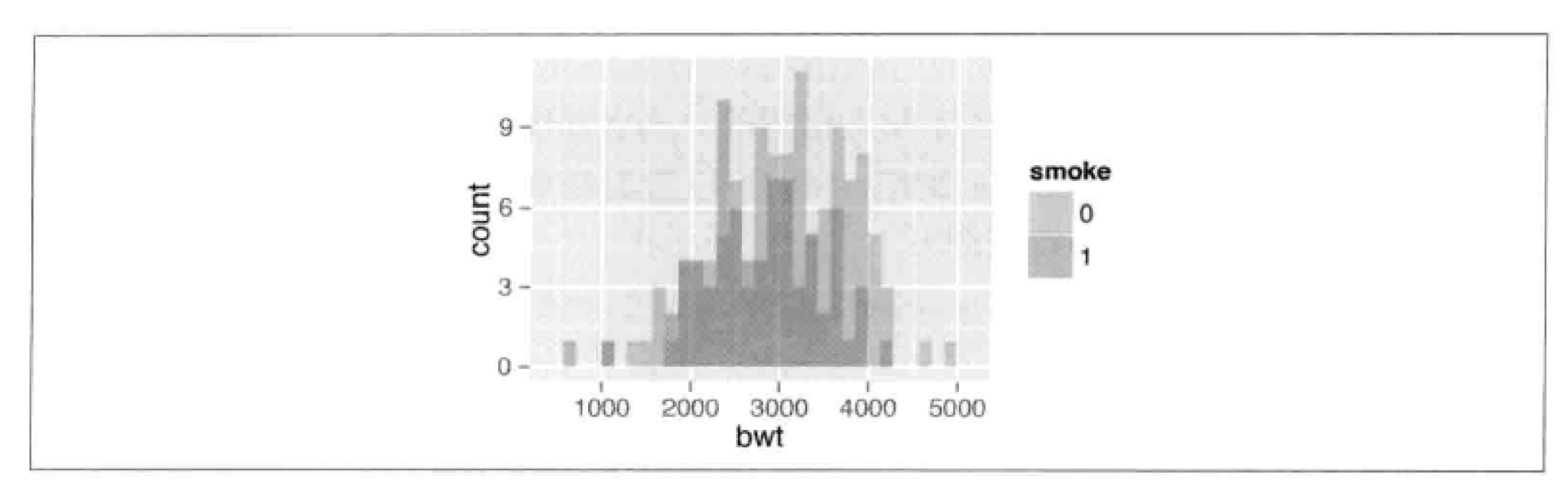


图 6-6 不同填充色的多组直方图

6.3 绘制密度曲线

问题

如何绘制核密度曲线？

方法

运行 geom_density() 函数，并映射一个连续型变量到 x（见图 6-7）：

```
ggplot(faithful, aes(x=waiting)) + geom_density()
```

如果不想绘制图形两侧和底部的线段，可以使用 geom_line(stat="density") 函数（见图 6-7 右图）：

```
# 使用 expand_limits() 函数扩大 y 轴范围以包含 0 点
ggplot(faithful, aes(x=waiting)) + geom_line(stat="density") +
    expand_limits(y=0)
```

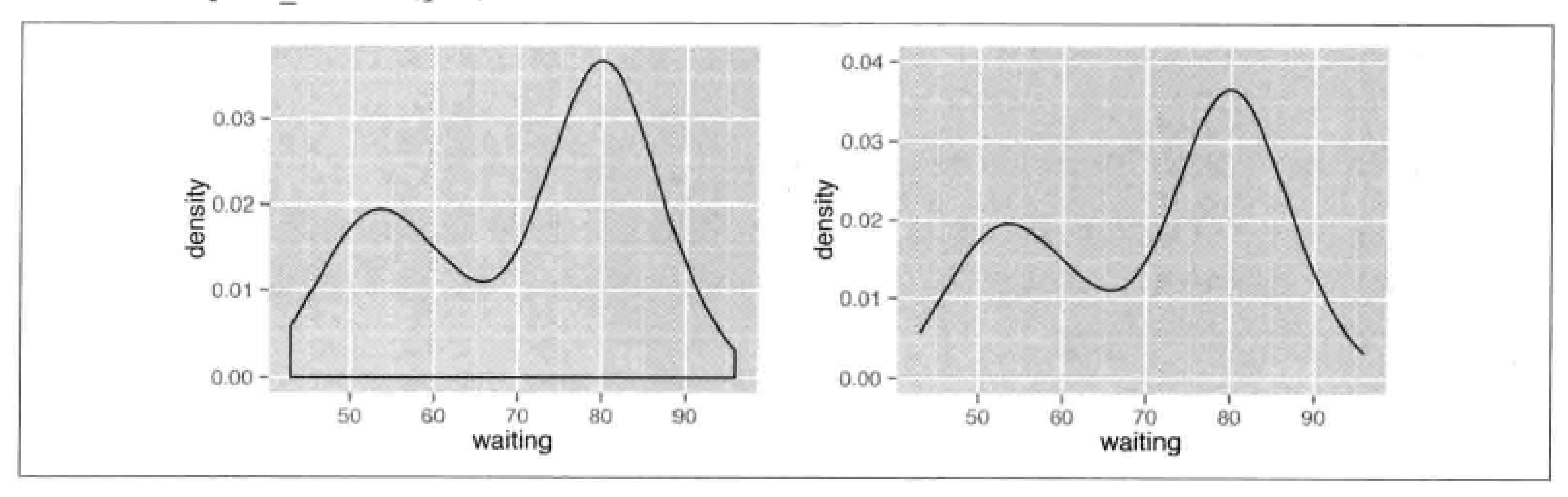


图 6-7 左图：**geom_density()** 函数绘制的核密度曲线 右图：**geom_line()** 函数绘制的核密度曲线

讨论

与 geom_histogram() 函数类似，geom_density() 函数只需要数据框中的一列作为参数。以 faithful 数据集为例，该数据包含了两列关于老忠实喷泉的数据：一列是 eruptions，表示

喷泉喷发的时长；第二列是waiting，表示两次喷发之间的时间间隔。下面仅以waiting列为例：

```
faithful

eruptions waiting
    3.600      79
    1.800      54
    3.333      74
...
```

上面提到的第二种方法是使用 geom_line() 函数，并告诉其使用 density 统计变换。这种方法与第一种使用 geom_density() 函数的方法在本质上是相同的，只不过前者绘制的是封闭的多边形。

与使用 geom_histogram() 函数类似，如果想快速地绘制未在数据框中的数据的直方图，可以在运行上述命令时，将数据框设定为 NULL，同时，向 ggplot() 函数传递一个包含所需数据的向量作为参数。这与第一种解决方案的结果相同：

```
# 将变量值保存在一个简单向量里
w <- faithful$waiting

ggplot(NULL, aes(x=w)) + geom_density()
```

核密度曲线是基于样本数据对总体分布做出的一个估计。曲线的光滑程度取决于核函数的带宽：带宽越大，曲线越光滑。带宽可以通过 adjust 参数进行设置，其默认值为 1。图 6-8 演示了当 adjust 更大或者更小时图形的展示效果：

```
ggplot(faithful, aes(x=waiting)) +
    geom_line(stat="density", adjust=.25, colour="red") +
    geom_line(stat="density") +
    geom_line(stat="density", adjust=2, colour="blue")
```

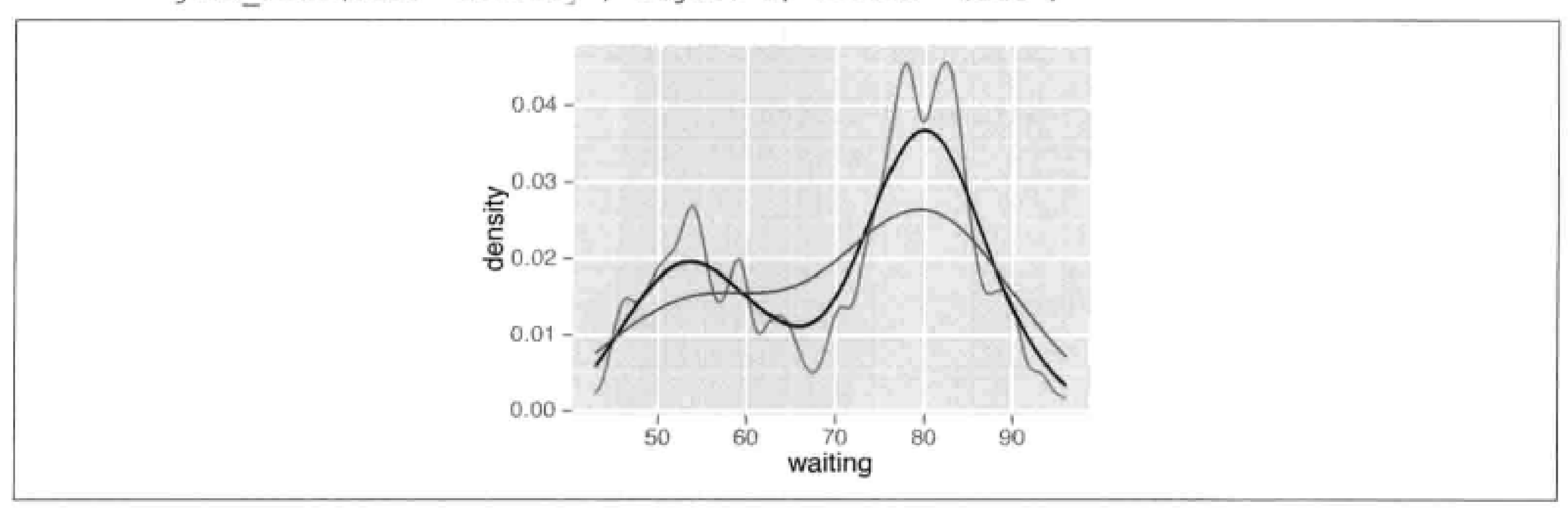


图 6-8　adjust 分别为 0.25（红线）、默认值 1（黑线）和 2（蓝线）的密度曲线

本例中，x 轴的坐标范围是自动设定的，以使其能包含相应的数据，但这会导致曲线的边缘被裁剪。想要展示曲线的更多部分，可以手动设定 x 轴的范围（见图 6-9）。同时，设置 alpha=.2 使填充色的透明度为 80%。

```
ggplot(faithful, aes(x=waiting)) +
    geom_density(fill="blue", alpha=.2) +
    xlim(35, 105)
```

```
# 这段代码将使用 geom_density() 函数绘制一个蓝色多边形，并在顶端添加一条实线
ggplot(faithful, aes(x=waiting)) +
    geom_density(fill="blue", colour=NA, alpha=.2) +
    geom_line(stat="density") +
    xlim(35, 105)
```

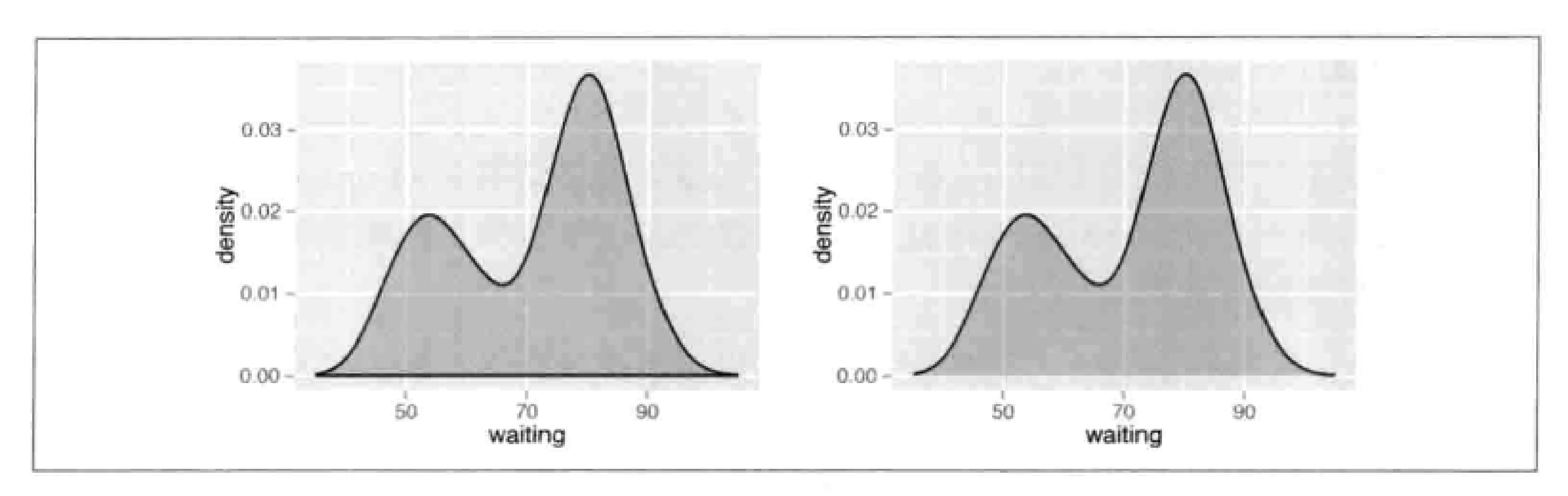


图 6-9　左图：*x* 轴范围更大、填充色为半透明的密度曲线　右图：geom_density() 和 geom_line() 共同绘制的密度曲线

如果数据集在绘图时发生了曲线边缘被裁剪的情况，那可能是因为相应的核密度曲线过于平滑——如果核密度曲线的宽度超过相应的数据集的范围，则其可能并非最好的模型。当然，也可能是因为数据集太小了。

将密度曲线叠加到直方图上，可以对观测值的理论分布和实际分布进行比较。由于密度曲线对应的 *y* 轴坐标较小（曲线下的面积总是 1），如果将其叠加到未做任何变换的直方图上，曲线可能会很难看清楚。通过设置 y=..density.. 可以减小直方图的标度以使其与密度曲线的标度相匹配。这里，我们先运行 geom_histogram() 函数绘制直方图，之后，运行 geom_density() 函数将密度曲线绘制到更上一层的图层上（见图 6-10）：

```
ggplot(faithful, aes(x=waiting, y=..density..)) +
    geom_histogram(fill="cornsilk", colour="grey60", size=.2) +
    geom_density() +
    xlim(35, 105)
```

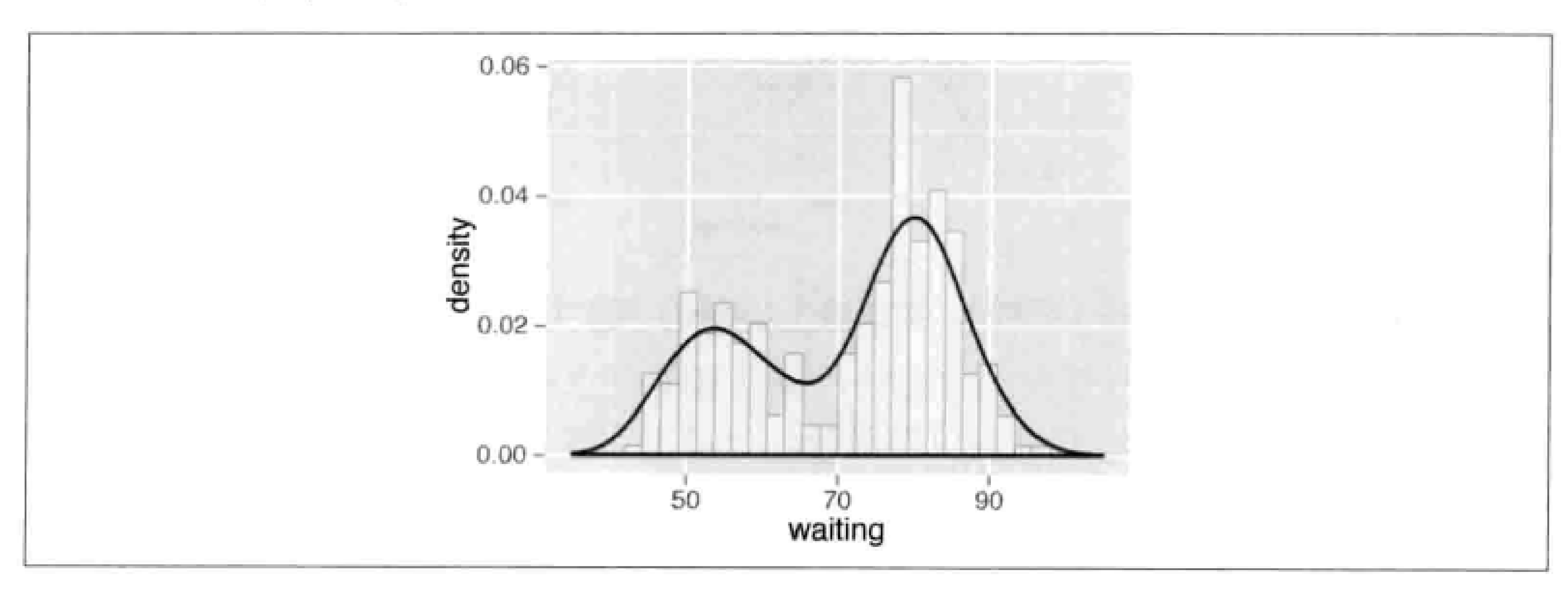


图 6-10　叠加到直方图上的密度曲线

另见

6.9 节介绍了小提琴图的相关内容，小提琴图是另一种表示密度曲线的方法，其适合用来对多个分布进行比较。

6.4 基于分组数据绘制分组密度曲线

问题

如何基于分组数据绘制分组密度曲线？

方法

使用 geom_density() 函数，将分组变量映射给 colour 或 fill 等图形属性即可，如图 6-11 所示。分组变量必须是因子型或者字符串向量。数据集 birthwt 对应的最佳分组变量 smoke 被存储为数值型，所以，我们必须先将其转化为因子：

```
library(MASS) # 为了使用数据
# 复制数据的副本
birthwt1 <- birthwt

# 把变量 smoke 转化为因子
birthwt1$smoke <- factor(birthwt1$smoke)

# 把变量 smoke 映射给 colour
ggplot(birthwt1, aes(x=bwt, colour=smoke)) + geom_density()

# 把变量 smoke 映射给 fill，设置 alpha 使填充色半透明
ggplot(birthwt1, aes(x=bwt, fill=smoke)) + geom_density(alpha=.3)
```

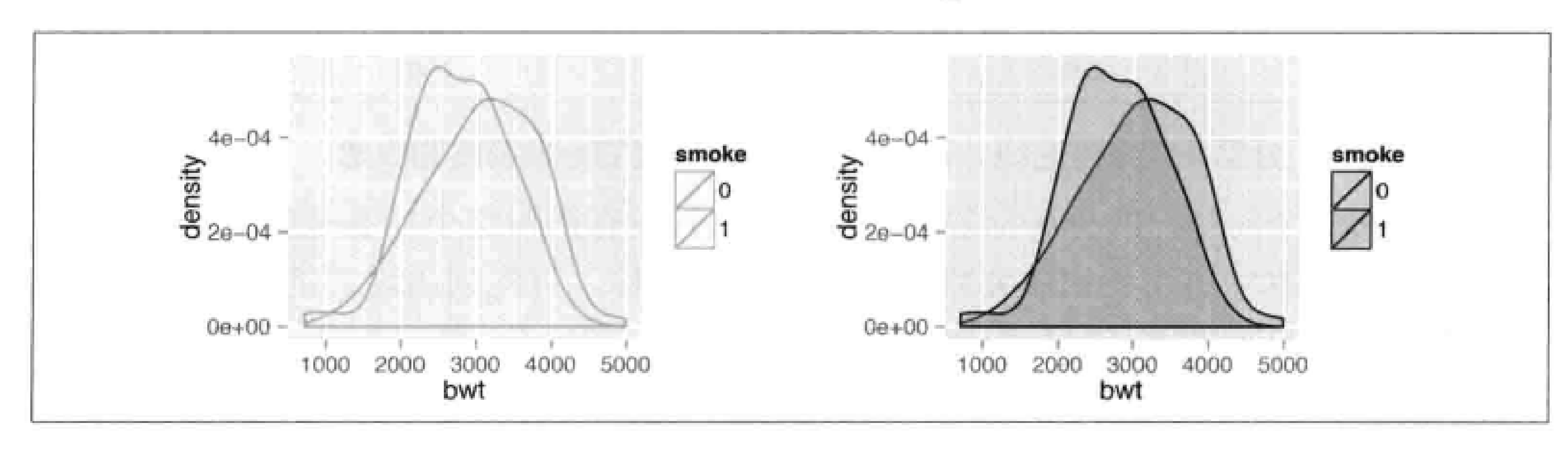


图 6-11　左图：每组数据对应于不同的线条颜色　右图：每组数据对应于不同的半透明填充色

讨论

绘制上图时，要求所有用到的数据都包含在一个数据框里，且数据框的其中一列是可用于分组的分类变量。

这里以 birthwt 数据集为例。该数据集包含的是关于婴儿出生体重及一系列导致出生体重过低的危险因子的数据：

```
birthwt

low age lwt race smoke ptl ht ui ftv  bwt
  0  19 182    2     0   0  0  1   0 2523
  0  33 155    3     0   0  0  0   3 2551
  0  20 105    1     1   0  0  0   1 2557
...
```

观察一下变量 smoke（抽烟与否）与变量 bwt（出生体重，单位是克）的关系。变量 smoke 对应的取值是 0 和 1，但由于其被存储为数值型向量，因而 ggplot() 函数不知道应当将其作为分类变量来处理。这时有两种方法可以选择，一是将数据框中相应的列转化为因子，二是通过在 aes() 函数内部使用命令 factor(smoke) 来告诉 ggplot() 函数把 smoke 当作因子来处理。本例中，我们将其转化为因子。

另一种对分组数据分布进行可视化的方法是使用分面（facet），如图 6-12 所示。可以令各个分面竖直对齐或者水平对齐。下面，我们将各分面竖直对齐来对分面中的两个分布进行比较：



```
ggplot(birthwt1, aes(x=bwt)) + geom_density() + facet_grid(smoke ~ .)
```

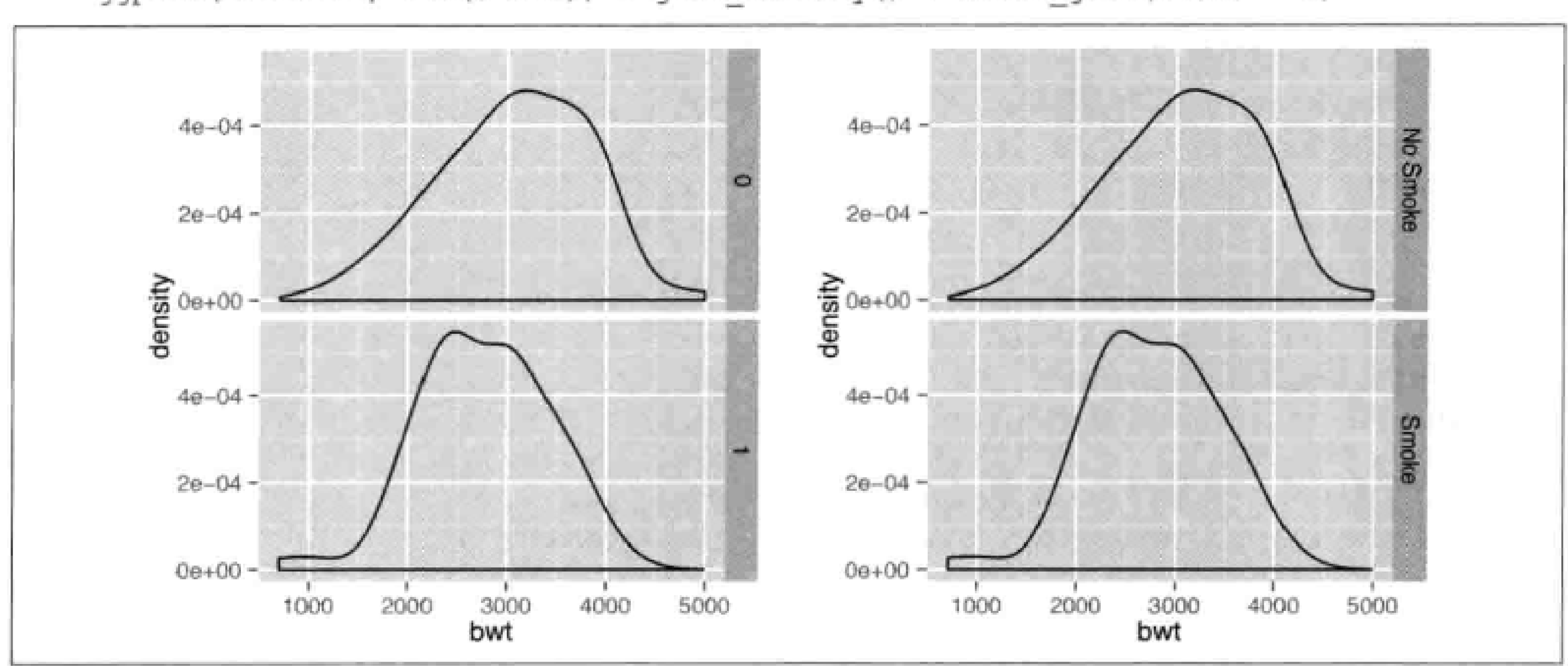


图 6-12　左图：分面绘制密度曲线　右图：修改分面标签的分面密度曲线图

分面绘图有一个问题，即分面标签只有 0 和 1，且没有指明这个标签是变量 smoke 的取值。想要修改标签，我们需要修改因子水平的名称。首先列出现有的因子水平，然后，依照相同的顺序向它们赋予新的名字：

```
levels(birthwt1$smoke)

"0" "1"

library(plyr) # 为了使用 revalue 函数
birthwt1$smoke <- revalue(birthwt1$smoke, c("0"="No Smoke", "1"="Smoke"))
```

重新绘图，图形中为新的分面标签（见图 6-12 右图）。

```
ggplot(birthwt1, aes(x=bwt)) + geom_density() + facet_grid(smoke ~ .)
```

如果要将直方图和密度曲线绘制在一张图上，最佳方案是利用分面，这是因为将两个

直方图绘制在同一张图上的其他方法都不易于解释。操作时，需设定 y=..density..，这样系统会将直方图的 y 轴标度降到跟密度曲线相同。在本例中，通过修改直方图颜色使直方图的条形不那么突出（见图 6-13）：

```
ggplot(birthwt1, aes(x=bwt, y=..density..)) +
    geom_histogram(binwidth=200, fill="cornsilk", colour="grey60", size=.2) +
    geom_density() +
    facet_grid(smoke ~ .)
```

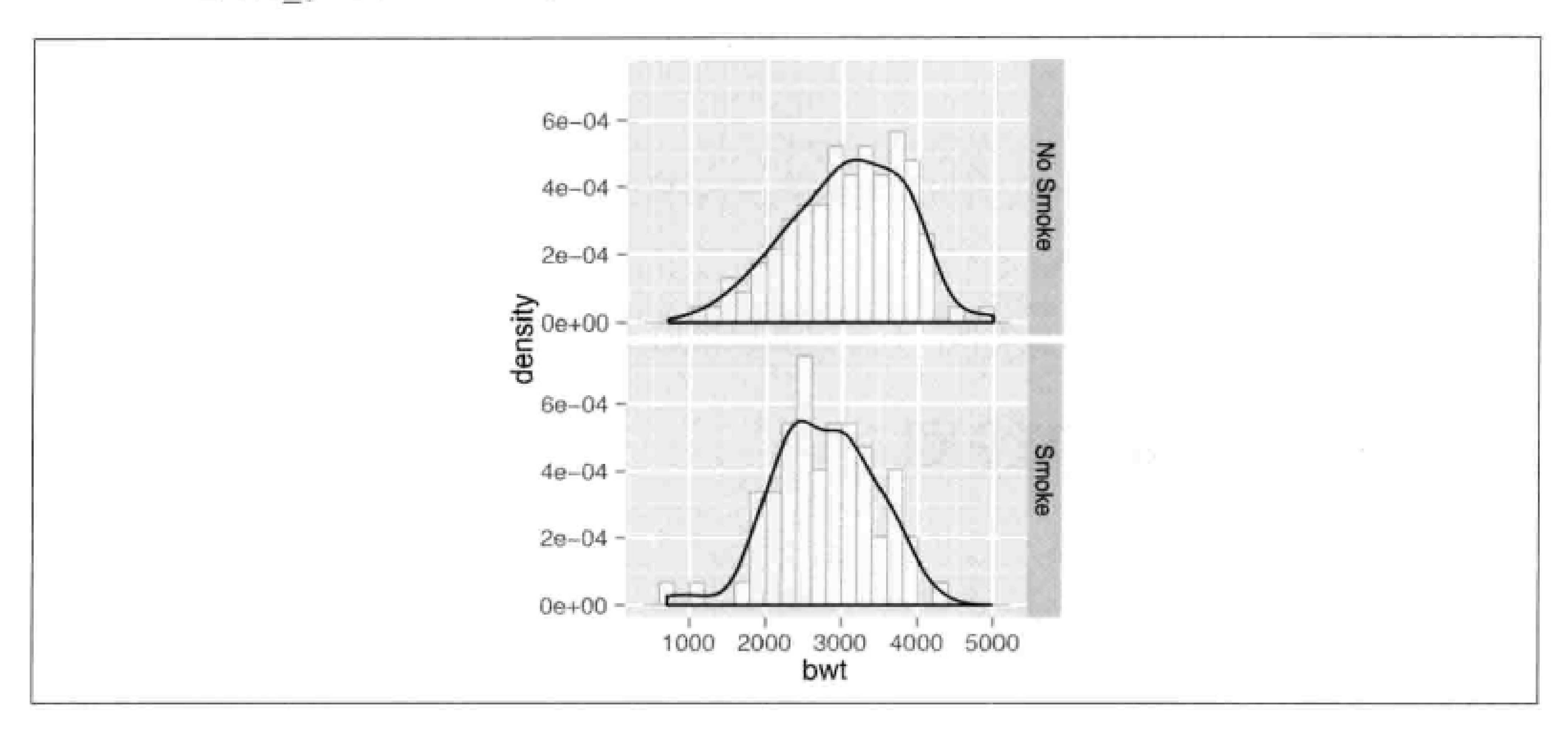


图 6-13　叠加在直方图上的密度曲线

6.5　绘制频数多边形

问题

如何绘制频数多边形？

方法

使用 geom_freqpoly() 函数即可（见图 6-14 左图）：

```
ggplot(faithful, aes(x=waiting)) + geom_freqpoly()
```

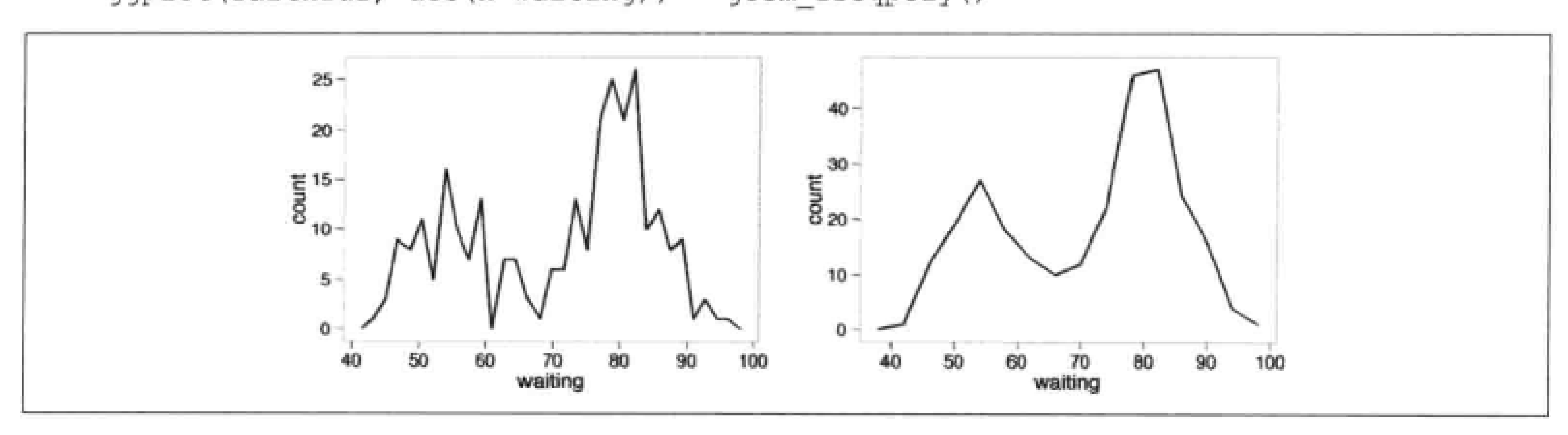


图 6-14　左图：频数多边形　右图：组距更大的频数多边形

讨论

频数多边形看起来跟核密度估计曲线相似，但其传递的信息类似于直方图。它跟直方图都描述了数据本身的信息，而核密度曲线只是一个估计，且需要人为输入带宽参数。

与直方图类似，可以通过 binwidth 参数控制频数多边形的组距（见图 6-14 右图）：

```
ggplot(faithful, aes(x=waiting)) + geom_freqpoly(binwidth=4)
```

或者，通过直接设定每组组距将数据的 *x* 轴范围切分为特定数目的组：

```
# 将组数设定为 15
binsize <-diff(range(faithful$waiting))/15
ggplot(faithful, aes(x=waiting)) + geom_freqpoly(binwidth=binsize)
```

另见

直方图与频数多边形传递的信息类似，只不过其以条形代替了直线。相关内容可参见 6.1 节。

6.6 绘制基本箱线图

问题

如何绘制箱线图（箱线胡须图）？

方法

使用 geom_boxplot() 函数，分别映射一个连续型变量和一个离散型变量到 y 和 x 即可（见图 6-15）：

```
library(MASS) # 为了使用数据集

ggplot(birthwt, aes(x=factor(race), y=bwt)) + geom_boxplot()
# 使用 factor() 函数将数值型变量转化为离散型
```

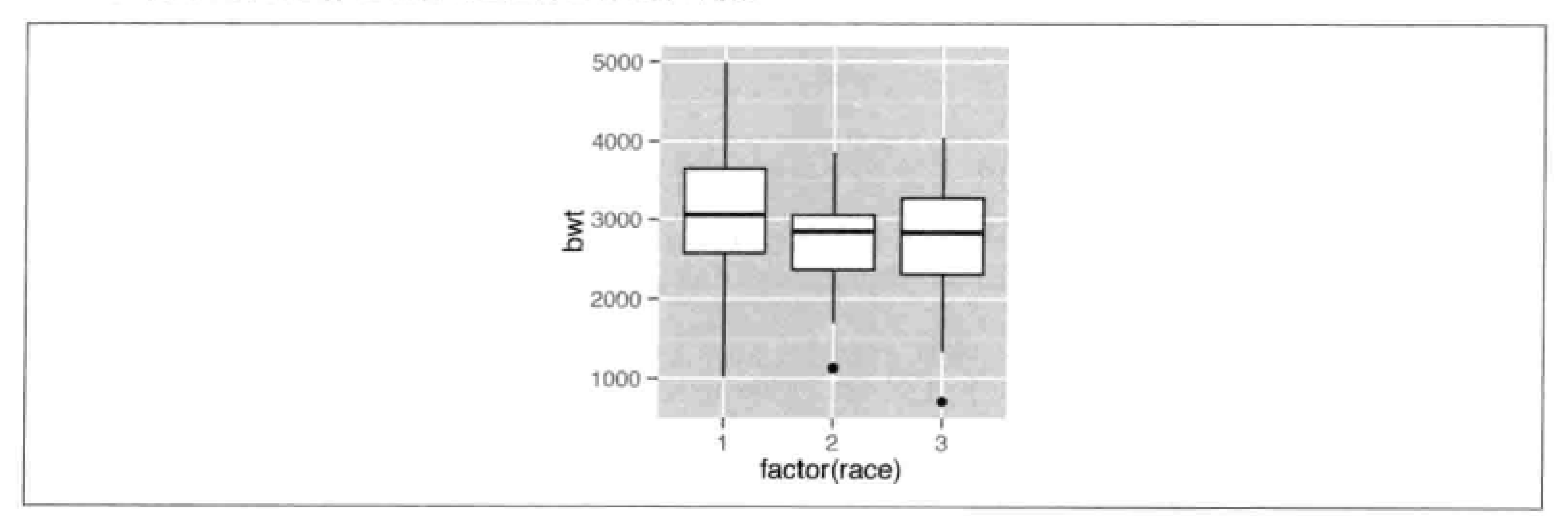


图 6-15 箱线图

讨论

下面以 MASS 库中的 birthwt 数据集为例。该数据集包含的是关于婴儿出生体重及一

系列导致出生体重过低的危险因子的数据：

```
birthwt

low age lwt race smoke ptl ht ui ftv  bwt
  0  19 182    2     0   0  0  1   0 2523
  0  33 155    3     0   0  0  0   3 2551
  0  20 105    1     1   0  0  0   1 2557
...
```

图 6-15 中，系统按变量 race 将数据分为三组，我们对每组数据对应的 bwt 变量进行可视化。变量 race 对应的值为 1、2、3，然而，由于其被存储为数值型向量，ggplot() 不知道如何将其当作分组变量来处理。为了使 ggplot() 能将其作为分组变量来处理，我们可以调整数据框把变量 race 转化为因子，或者通过在 aes() 语句内部使用 factor(race) 告诉 ggplot() 函数把 race 当作因子来处理。在前面的例子中，用的是 factor(race)。

箱线图由箱和“须”（whisker）两部分组成。箱的范围是从数据的下四分位数到上四分位数，也就是常说的四分位距（IQR）。箱的中间有一条表示中位数，或者说 50% 分位数的线。须则是从箱子的边缘出发延伸至 1.5 倍四分位距内的最远的点。如果图中有超过须的数据点，则其被视为异常值，并以点来表示。图 6-16 使用偏态的数据展示了直方图、密度曲线和箱线图之间的关系。

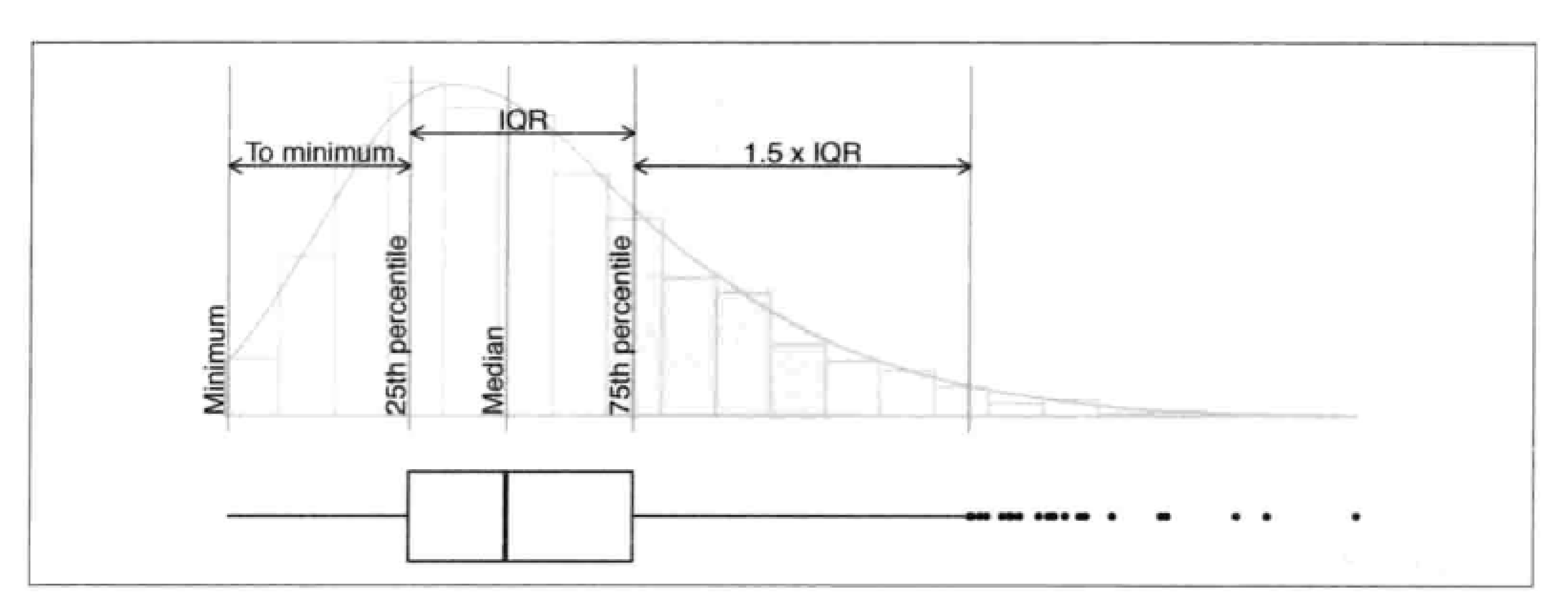


图 6-16　箱线图、直方图和密度曲线的对比图

设定参数 width 可以修改箱线图的宽度（见图 6-17 左图）：

```
ggplot(birthwt, aes(x=factor(race), y=bwt)) + geom_boxplot(width=.5)
```

如果图中异常值较多且图形有重叠的话，可以通过设置 outlier.size 和 outlier.shape 参数修改异常点的大小和点形。异常点默认的大小是 2，点形是 16，即实心圆（见图 6-17 右图）：

```
ggplot(birthwt, aes(x=factor(race), y=bwt)) +
    geom_boxplot(outlier.size=1.5, outlier.shape=21)
```

绘制单组数据的箱线图时，必须给 x 参数映射一个特定的取值，否则，ggplot() 函数不知道箱线图对应的 *x* 轴坐标。本例中，我们将其设定为 1，并移除 *x* 轴的刻度标记

（tick marker）和标签（见图 6-18）：

```
ggplot(birthwt, aes(x=1, y=bwt)) + geom_boxplot() +
    scale_x_continuous(breaks=NULL)+
    theme(axis.title.x = element_blank())
```

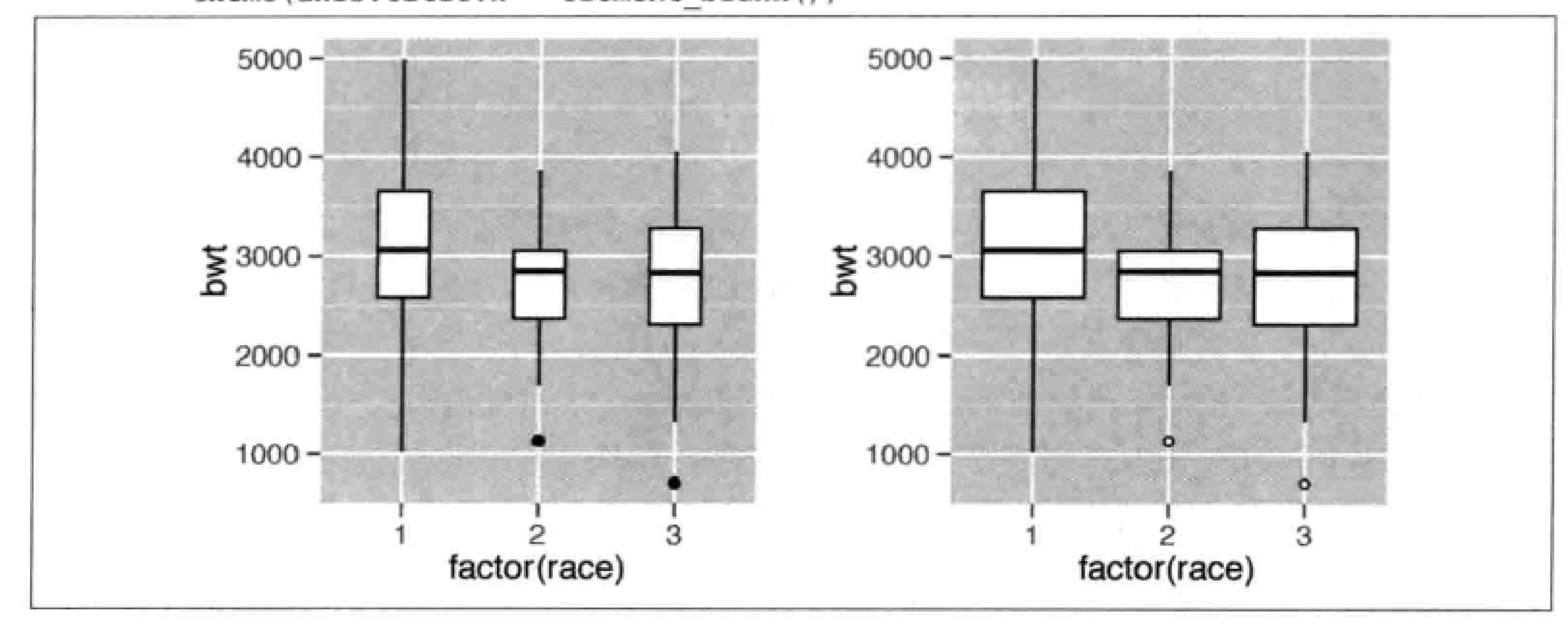


图 6-17　左图：箱子较窄的箱线图　右图：用小号空心圆表示的异常点

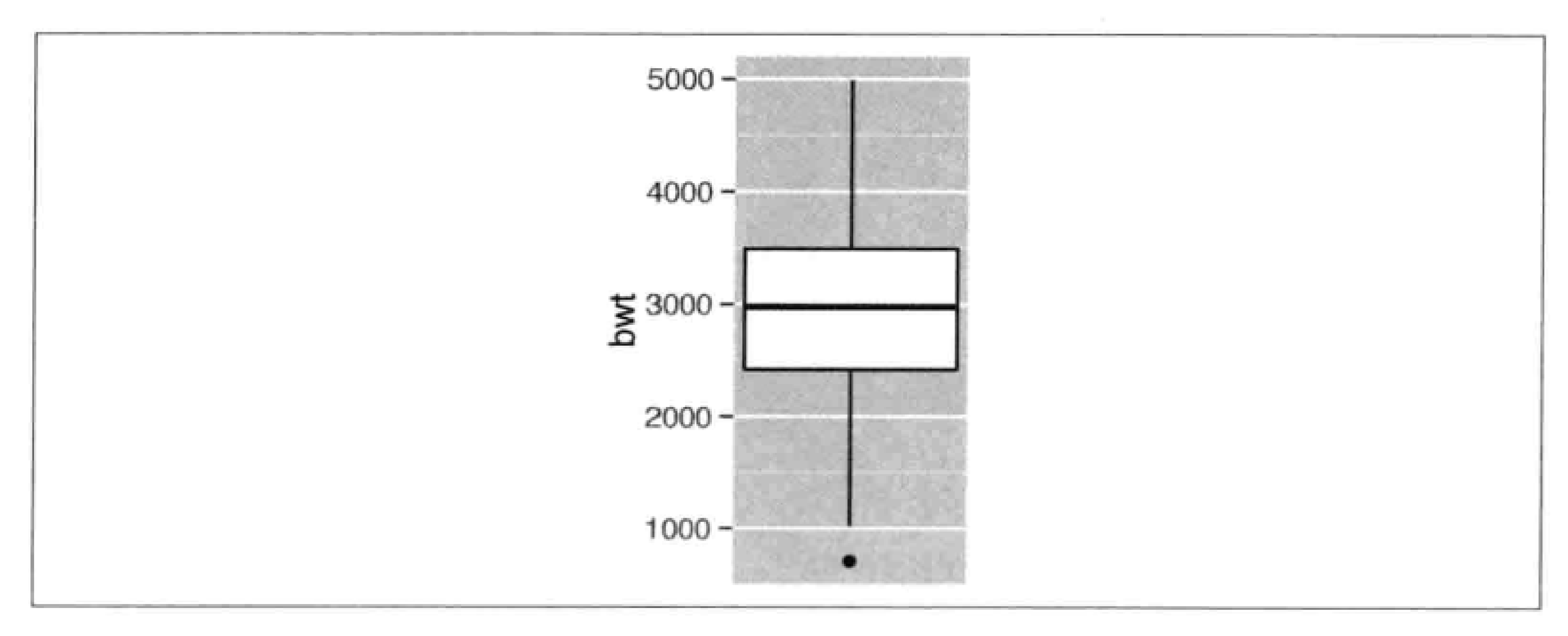


图 6-18　单组数据箱线图



这里计算分位数的方法与 R base 包中的 boxplot() 函数所使用的计算方法略有不同。当样本量较小时，这个差异可能会比较明显。键入 ?geom_boxplot() 命令可以查看这两种计算方法的差异。

6.7　向箱线图添加槽口

问题

如何向箱线图添加槽口（notch）以比较各组数据的中位数是否有差异？

方法

使用 geom_boxplot() 函数并设定参数 notch=TRUE（见图 6-19）：

```
library(MASS) # 为了使用数据
ggplot(birthwt, aes(x=factor(race), y=bwt)) + geom_boxplot(notch=TRUE)
```

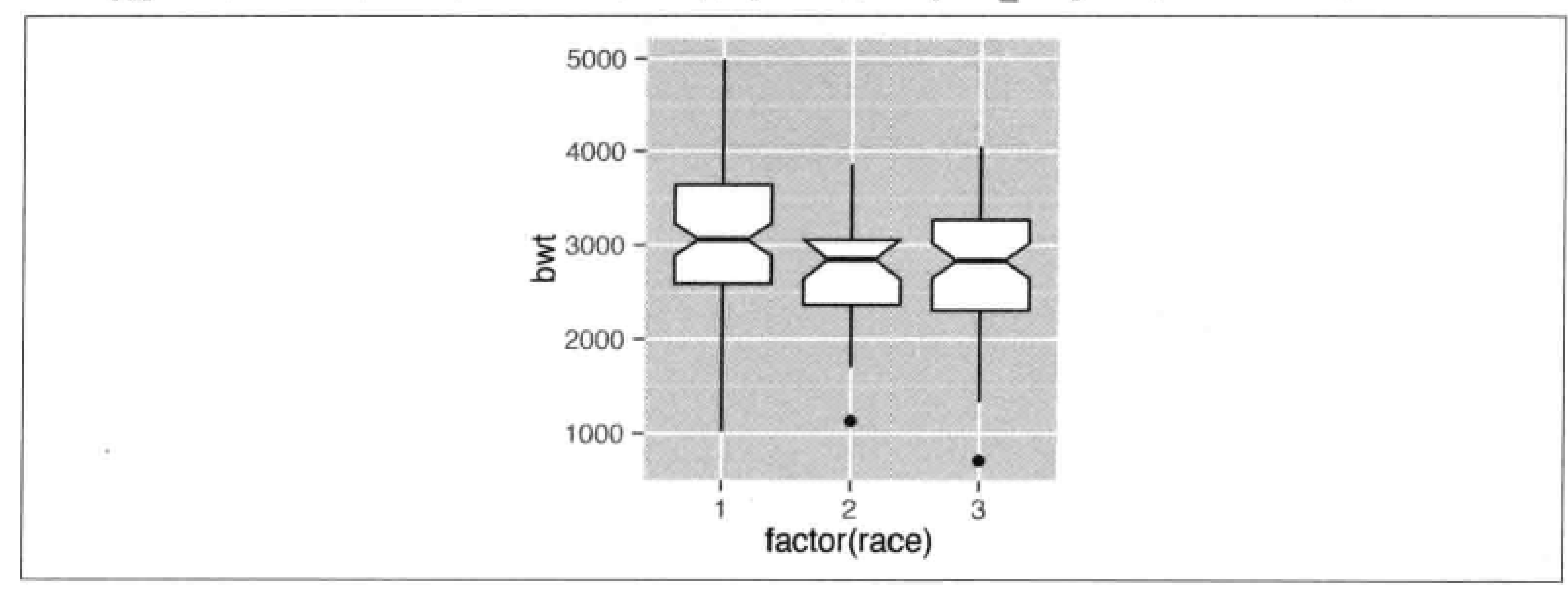


图 6-19　带槽口的箱线图

讨论

箱线图中的槽口用来帮助查看不同分布的中位数是否有差异。如果各箱线图的槽口互不重合，说明各中位数有差异。

对于本数据集，你会看到如下信息：

```
Notch went outside hinges. Try setting notch=FALSE.
```

这表明置信域（槽口）超过了某个箱子的边界。本例中，中间箱子对应的槽口的上边界溢出箱体，但由于溢出的距离较小，因此，在最终的绘图输出中几乎看不到。槽口溢出到箱体的边界并没有什么实质错误，只是在一些极端案例中会看起来很奇怪。

6.8　向箱线图添加均值

问题

如何向箱线图添加均值标记？

方法

使用 stat_summary() 函数。箱线图中的均值常以钻石形状来表示，所以，下面用点形 23 且填充色为白色的点来表示。同时，通过设置参数 size=3 使用略大的点（见图 6-20）：

```
library(MASS) # 为了使用数据

ggplot(birthwt, aes(x=factor(race), y=bwt)) + geom_boxplot() +
    stat_summary(fun.y="mean", geom="point", shape=23, size=3, fill="white")
```

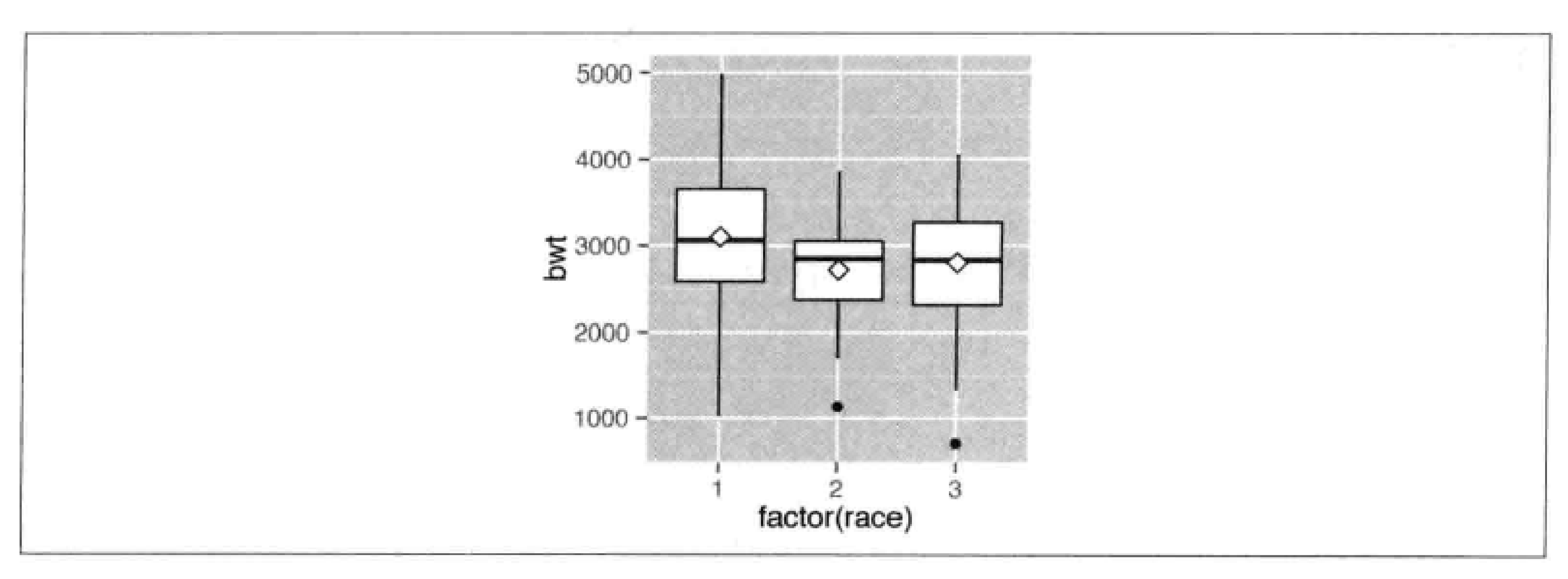


图 6-20　箱线图上的均值标记

讨论

箱线图中间的水平线表示的是中位数，而不是均值。对于正态分布的数据，中位数与均值会比较接近，但对于偏态的数据它们将有所不同。

6.9　绘制小提琴图

问题

如何绘制小提琴图以对各组数据的密度估计进行比较？

方法

使用 geom_violin() 函数即可（见图 6-21）：

```
library(gcookbook) # 为了使用数据

# 简单绘图
p <-ggplot(heightweight, aes(x=sex, y=heightIn))

p + geom_violin()
```

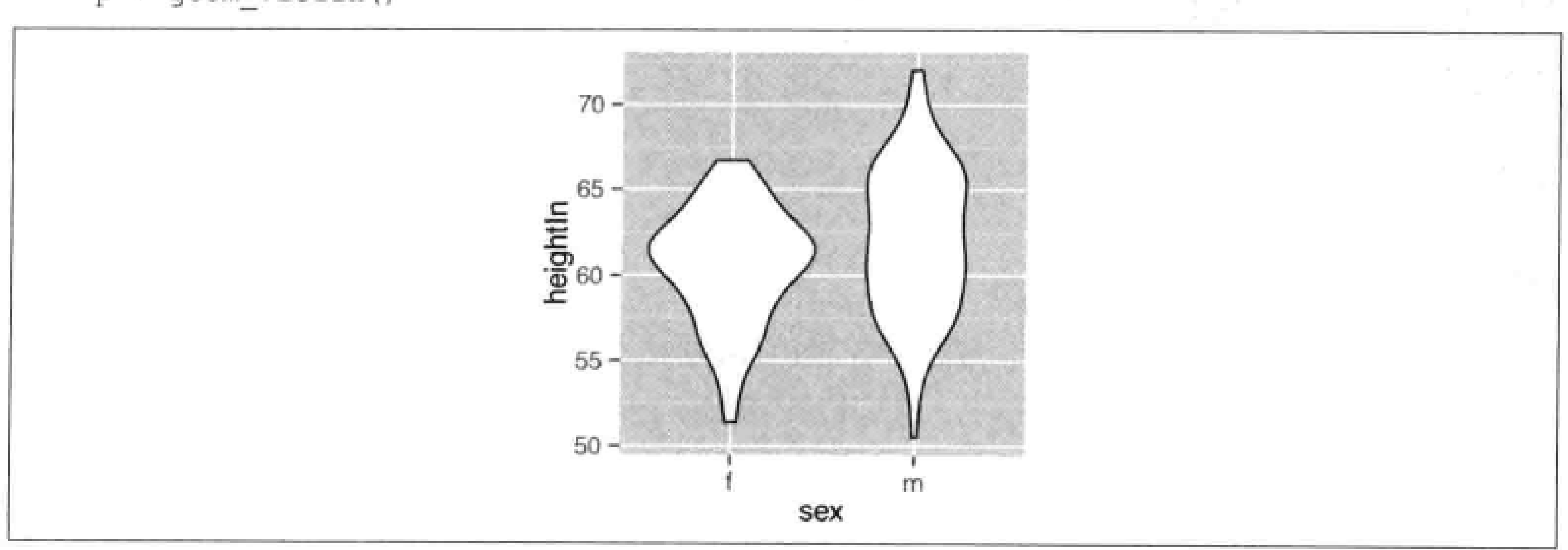


图 6-21　小提琴图

讨论

小提琴图是一种用来对多组数据的分布进行比较的方法。使用普通的密度曲线对多组数据进行可视化时，图中各曲线会彼此干扰，因而，不宜用来对多组数据的分布进行比较。而小提琴图是并排排列的，用它对多组数据的分布进行比较会更容易一些。

小提琴图也是核密度估计，但绘图时对核密度曲线取了镜像以使形状对称。传统画法中，小提琴图中间叠加了一个较窄的箱线图，同时，用一个白圆圈表示中位数，如图 6-22 所示。另外，通过设置 outlier.colour=NA 可以隐去箱线图中的异常点。

```
p + geom_violin() + geom_boxplot(width=.1, fill="black", outlier.colour=NA)+
    stat_summary(fun.y=median, geom="point", fill="white", shape=21, size=2.5)
```

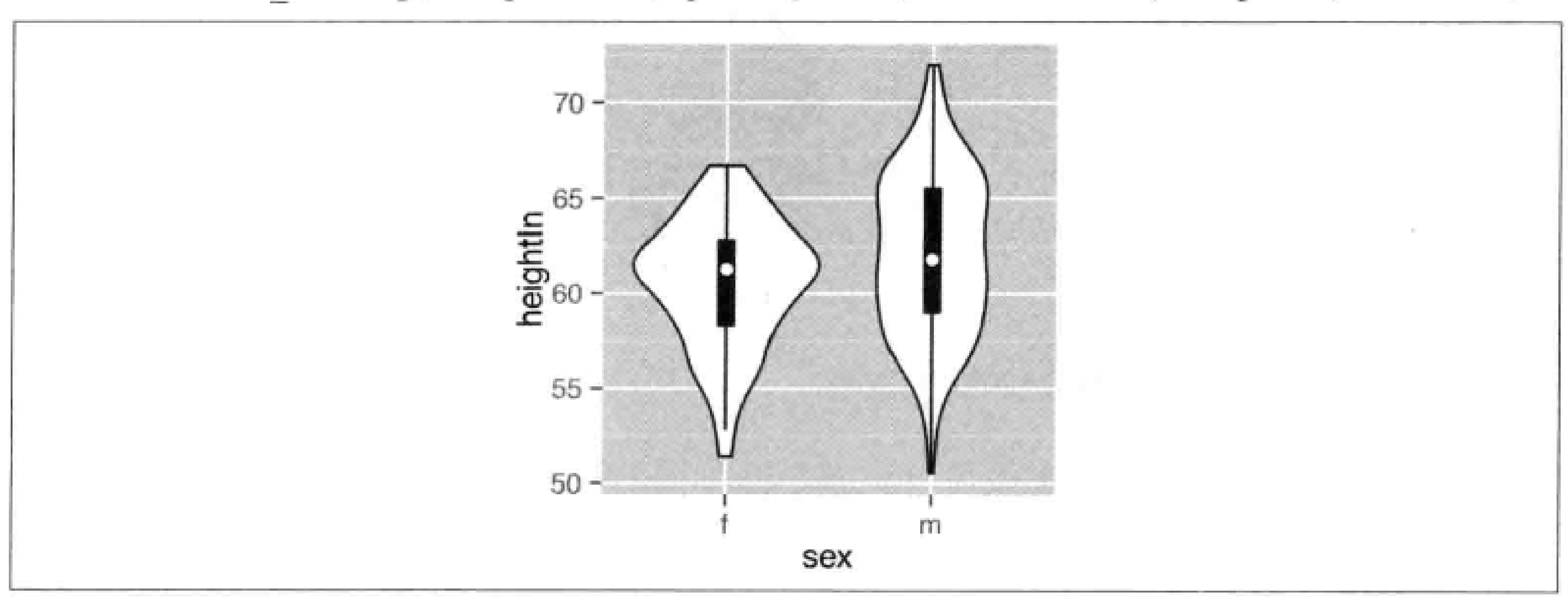


图 6-22　叠加箱线图的小提琴图

本例中，我们从下而上逐层绘制图形，先绘制小提琴图，再叠加箱线图，之后使用 stat_summary() 计算并绘制表示中位数的白圆圈。

小提琴图默认的坐标范围是数据的最小值到最大值，其扁平的尾部在这两个位置处截断。通过设置 trim=FALSE 可以保留小提琴的尾部（见图 6-23）：

```
p + geom_violin(trim=FALSE)
```

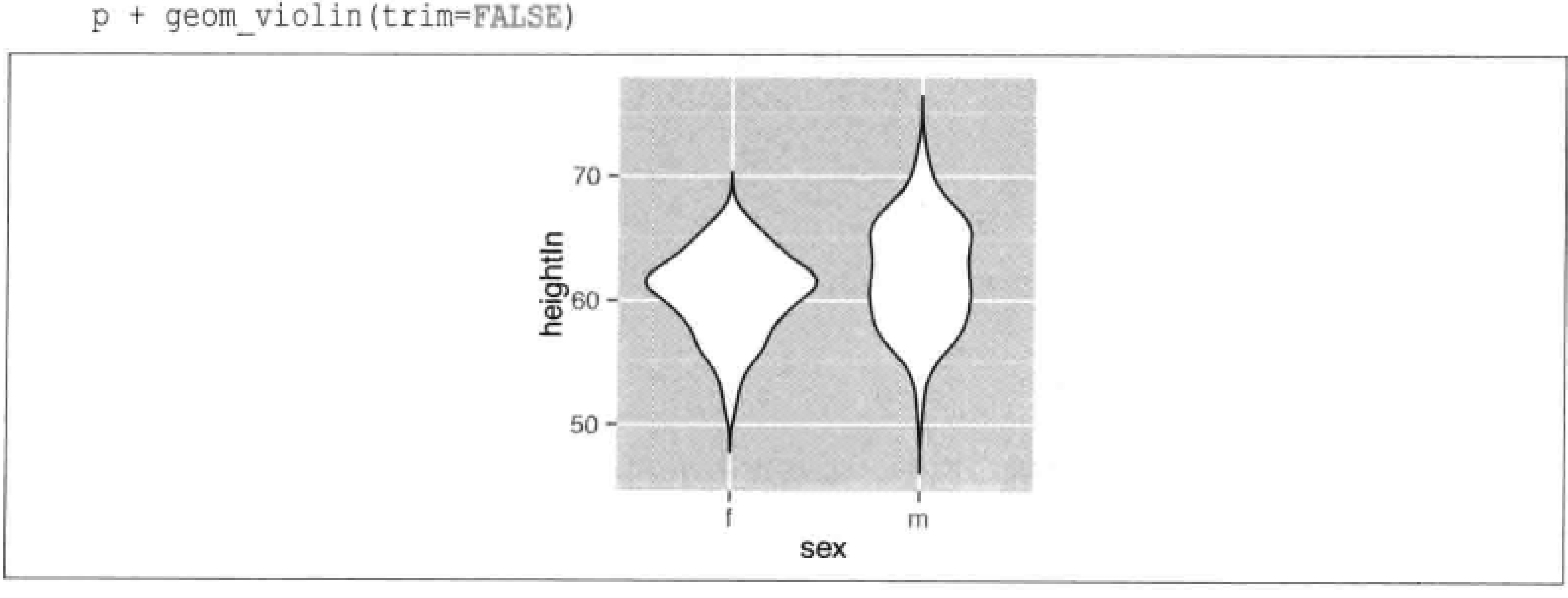


图 6-23　带尾部的小提琴图

默认情况下，系统会对小提琴图进行标准化以使得各组数据对应的图的面积一样（如果 trim=TRUE，对数据进行标准化时会包括尾部数据）。如果不想使各组数据对应的图的面积一样，可以通过设置 scale="count" 使得图的面积与每组观测值数目成正比（见图 6-24）。本例中，女性组数据比男性组数据略少，所以，f 组的小提琴图看起来略窄：

```
# 校准小提琴图的面积，令其与每组观测值的数目成正比
p + geom_violin(scale="count")
```

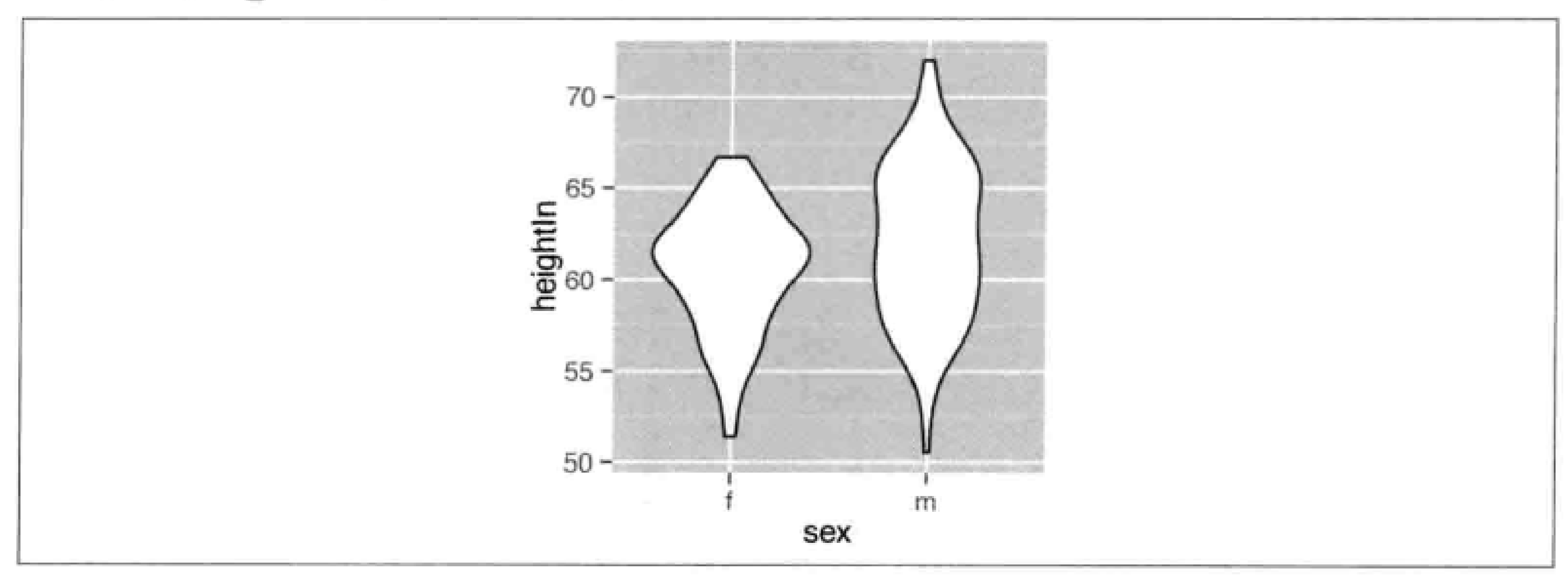


图 6-24　面积大小正比于观测数目的小提琴图

使用 6.3 节中介绍的 adjust 参数可以调整小提琴图的平滑程度。该参数的默认值是 1；更大的值对应于更平滑的曲线，反之亦然（见图 6-25）：

```
# 更平滑
p + geom_violin(adjust=2)

# 欠平滑
p + geom_violin(adjust=.5)
```

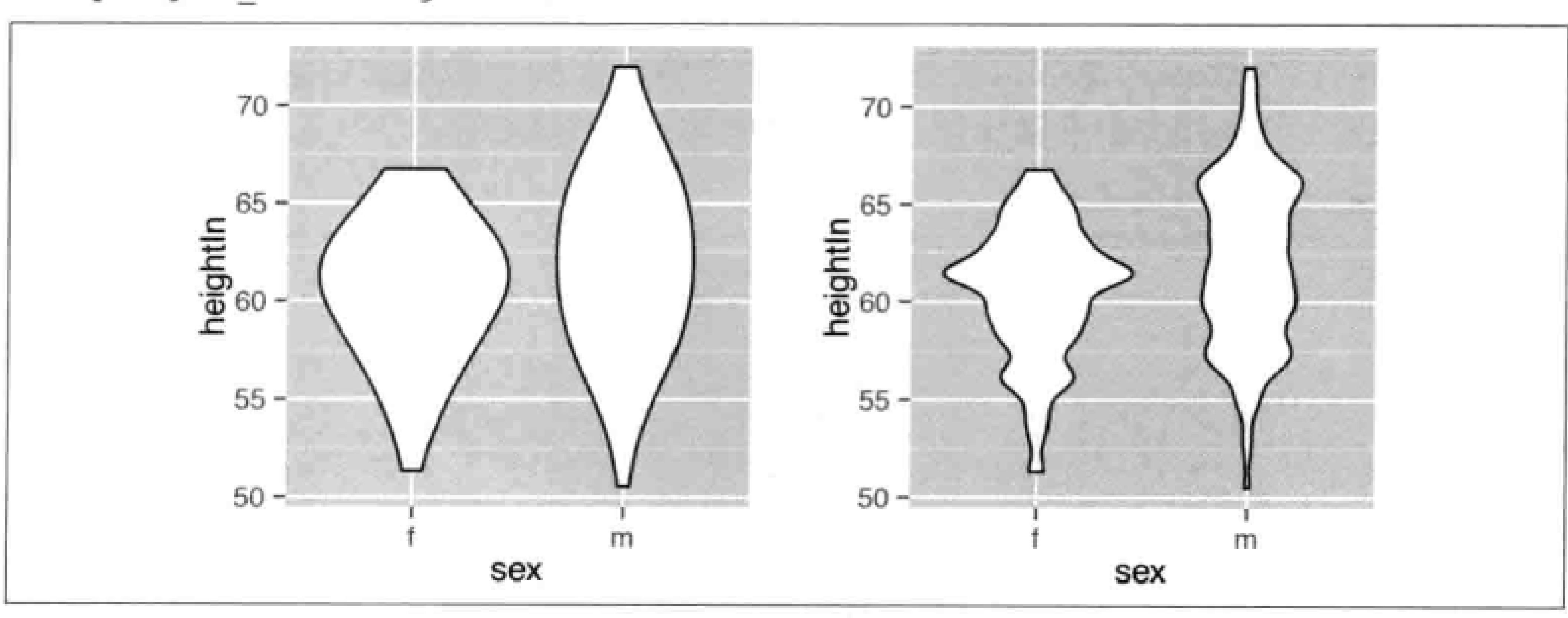


图 6-25　左图：更平滑的小提琴图　右图：平滑度更小的小提琴图

另见

创建传统密度曲线，可参考 6.3 节的内容。 使用不同于默认设置的点形，可参考 4.5

节的内容。

6.10 绘制 Wilkinson 点图

问题

如何绘制 Wikinson 点图，以展示所有数据点？

方法

使用 geom_dotplot() 函数。这里（见图 6-26）以数据集 countries 的子集为例：

```
library(gcookbook) # 为了使用数据
countries2009 <- subset(countries, Year==2009 & healthexp>2000)

p <- ggplot(countries2009, aes(x=infmortality))

p + geom_dotplot()
```

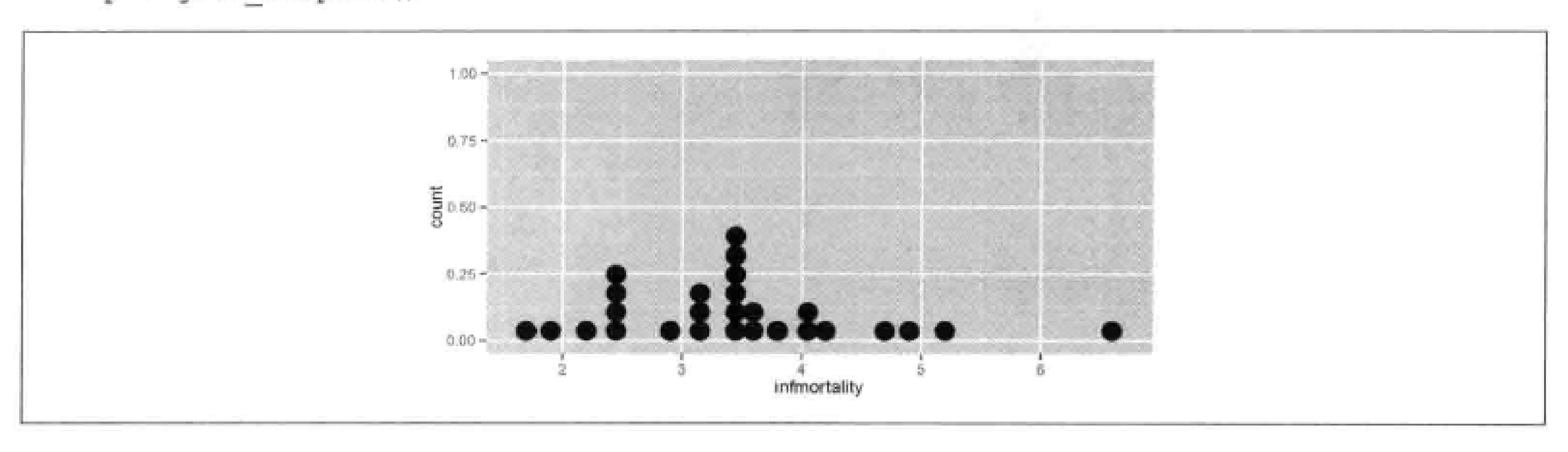


图 6-26 点图

讨论

这种点图有时又叫 Wilkinson 点图。它与 3.10 节中提到的 Cleveland 点图不同。这种图中，点的分组和排列取决于数据，每个点的宽度对应了最大的组距。系统默认的最大组距是数据范围的 1/30，我们可以通过 binwidth 参数对其进行调整。

默认情况下，geom_dotplot() 函数沿着 x 轴方向对数据进行分组，并在 y 轴方向上对点进行堆积。图中各点看起来是堆积的，但受限于 ggplot2 的技术，图形上 y 轴的刻度线没有明确的含义。使用 scale_y_continuous() 函数可以移除 y 轴标签。本例中，我们还使用 geom_rug() 函数以标示数据点的具体位置（见图 6-27）：

```
p + geom_dotplot(binwidth=.25) + geom_rug() +
    scale_y_continuous(breaks=NULL) +   # 移除刻度线
    theme(axis.title.y=element_blank()) # 移除坐标轴标签
```

你可能会注意到数据堆在水平方向上不是均匀分布的。根据默认的 dotdensity 分组算法，每个数据堆都放置在它表示的数据点的中心位置。要使用像直方图那样的固定间距的分组算法，可以令 method="histodot"。图 6-28 中，你将会发现图中的数据堆

并不是居中放置的。

```
p + geom_dotplot(method="histodot", binwidth=.25) + geom_rug() +
    scale_y_continuous(breaks=NULL) + theme(axis.title.y=element_blank())
```

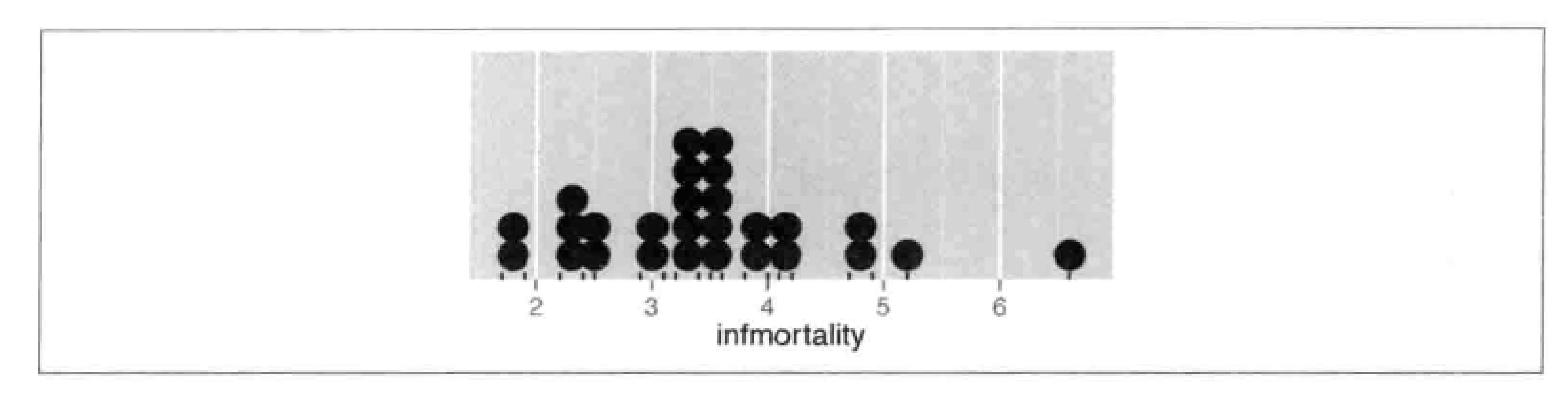


图 6-27　移除 *y* 轴标签、最大组距为 0.25 并添加边际地毯以标示数据点位置的点图

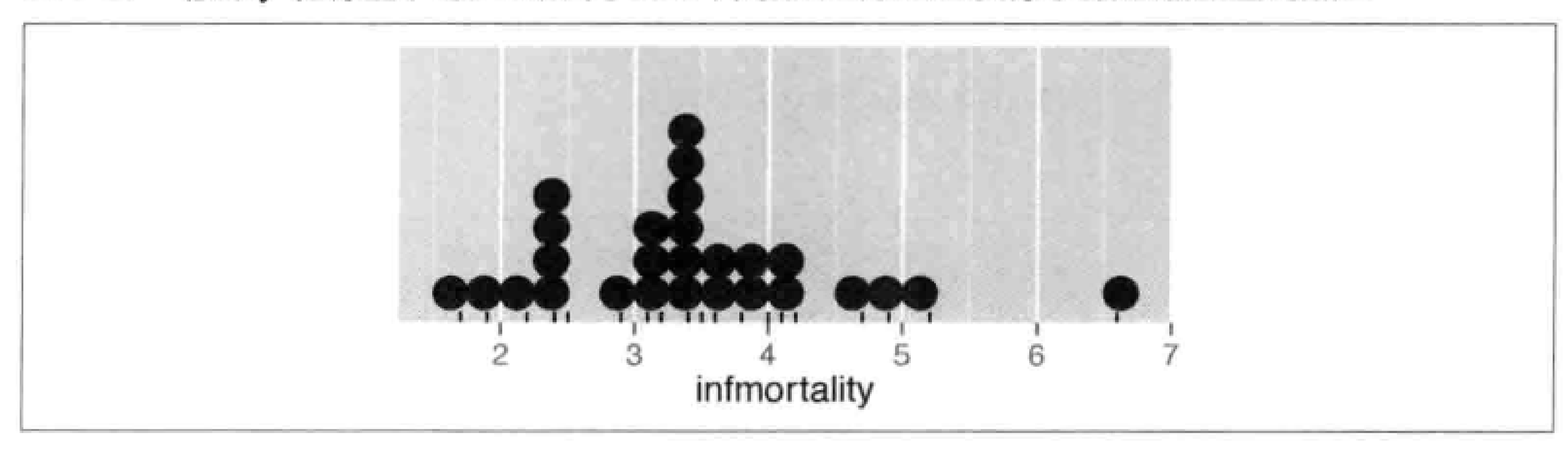


图 6-28　histodot 分组（固定宽度）时的点图

点图也能进行中心堆叠，或者采用一种奇数与偶数数量保持一致的中心堆叠方式。这可以通过设置 stackdir="center" 或者 stackdir="centerwhole" 来完成，如图 6-29 所示：

```
p + geom_dotplot(binwidth=.25, stackdir="center") +
    scale_y_continuous(breaks=NULL) + theme(axis.title.y=element_blank())

p + geom_dotplot(binwidth=.25, stackdir="centerwhole") +
    scale_y_continuous(breaks=NULL) + theme(axis.title.y=element_blank())
```

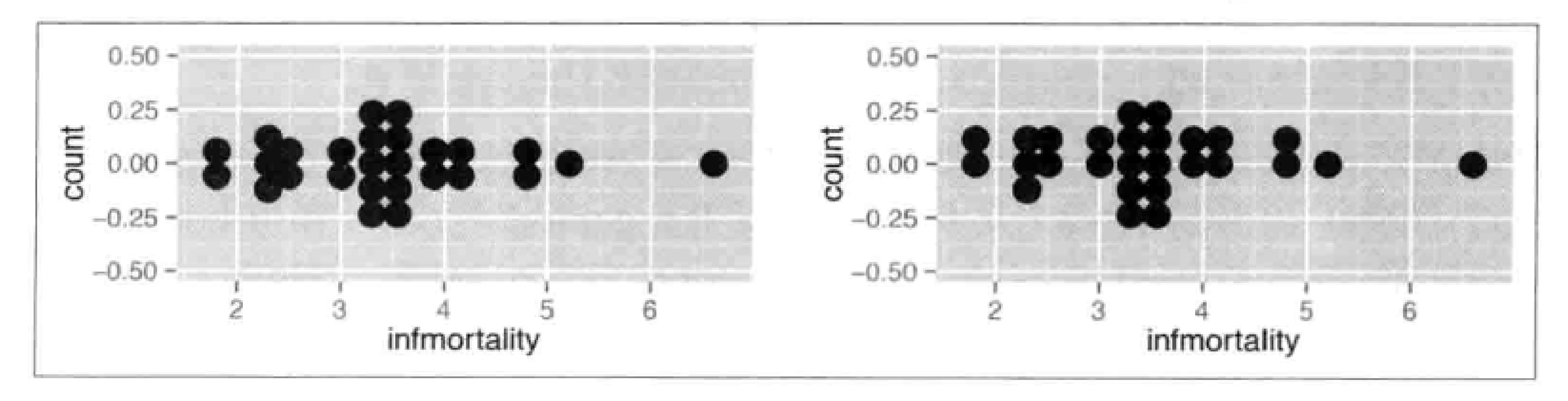


图 6-29　左图：stackdir="center" 时的点图　右图：stackdir="centerwhole" 时的点图

另见

Leland Wilkinson, "Dot Plots," The American Statistician 53 (1999): 276–281, http://www.cs.uic.edu /~wilkinson/Publications/dots.pdf.

6.11 基于分组数据绘制分组点图

问题

如何基于分组数据绘制多个点图？

方法

为了比较多组数据，可以通过设定 binaxis="y" 将数据点沿着 y 轴进行堆叠，并沿着 x 轴分组。本例中，我们将以 heightWeight 数据集为例（见图 6-30）：

```
library(gcookbook) # 为了使用数据

ggplot(heightweight, aes(x=sex, y=heightIn)) +
    geom_dotplot(binaxis="y", binwidth=.5, stackdir="center")
```

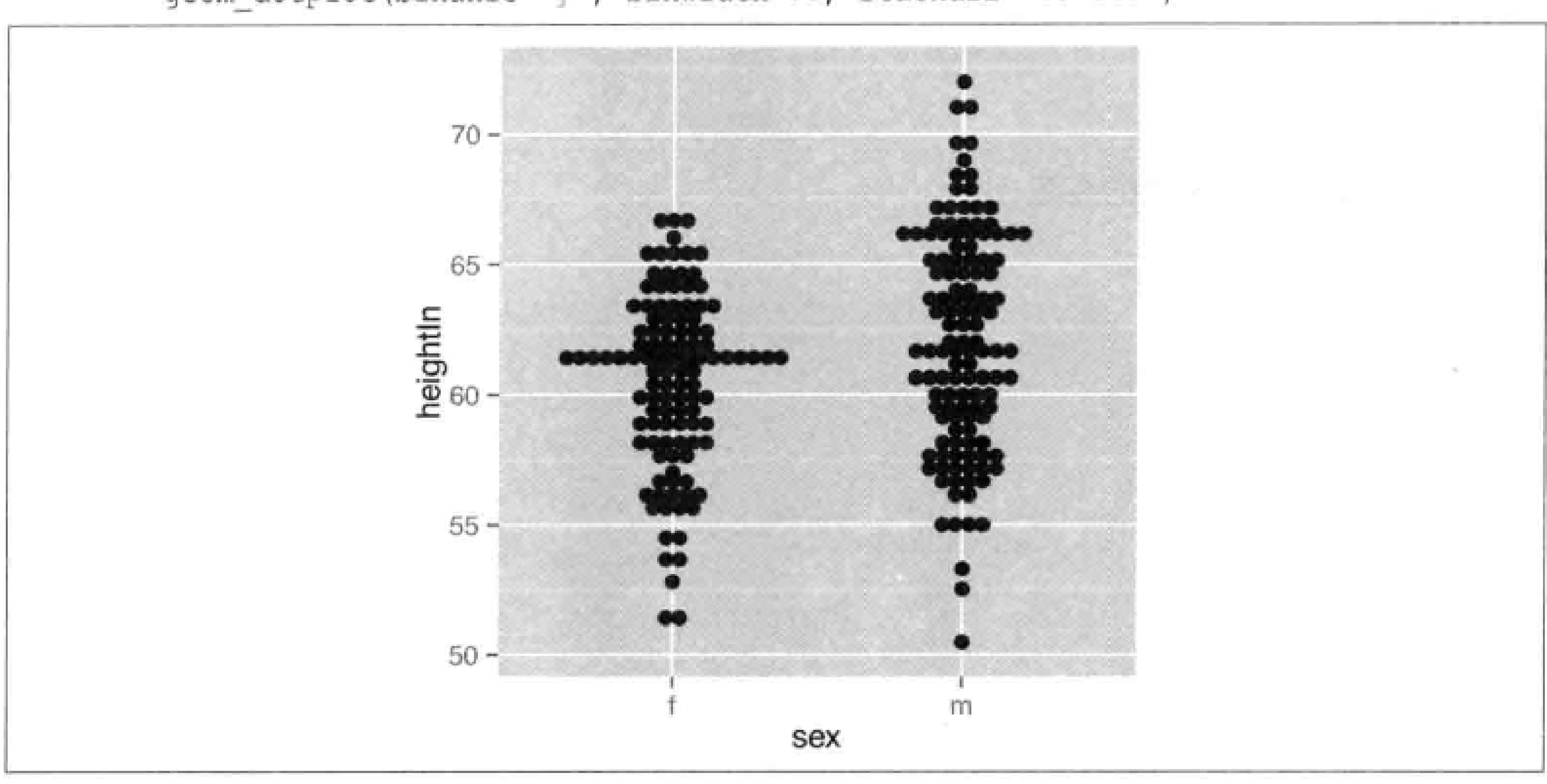


图 6-30　沿着 x 轴分组的多组数据点图

讨论

有时，我们会将点图叠加在箱线图上。这种情况下，应该将数据点变为空心，同时隐去箱线图上的异常点。这是因为异常点将作为点图的一部分展示出来（见图 6-31）：

```
ggplot(heightweight, aes(x=sex, y=heightIn)) +
    geom_boxplot(outlier.colour=NA, width=.4) +
    geom_dotplot(binaxis="y", binwidth=.5, stackdir="center", fill=NA)
```

也可以将点图置于箱线图旁边，如图 6-32 所示。这需要用到一些技巧：通过将 x 变量视作数值型变量并对其加减一个微小的数值移动箱线图和点图的位置，使点图位于箱线图的左边。

当 x 变量被视为数值型变量时，必须同时指定 group，否则，数据会被视为单独一组，从而，只绘制出一个箱线图和点图。最后，由于 x 轴被视为数值型，系统会默认展示

x 轴刻度标签的数值；必须通过 `scale_x_continuous()` 函数对其进行调整，以使得 x 轴的刻度标签显示为与因子水平相对应的文本：

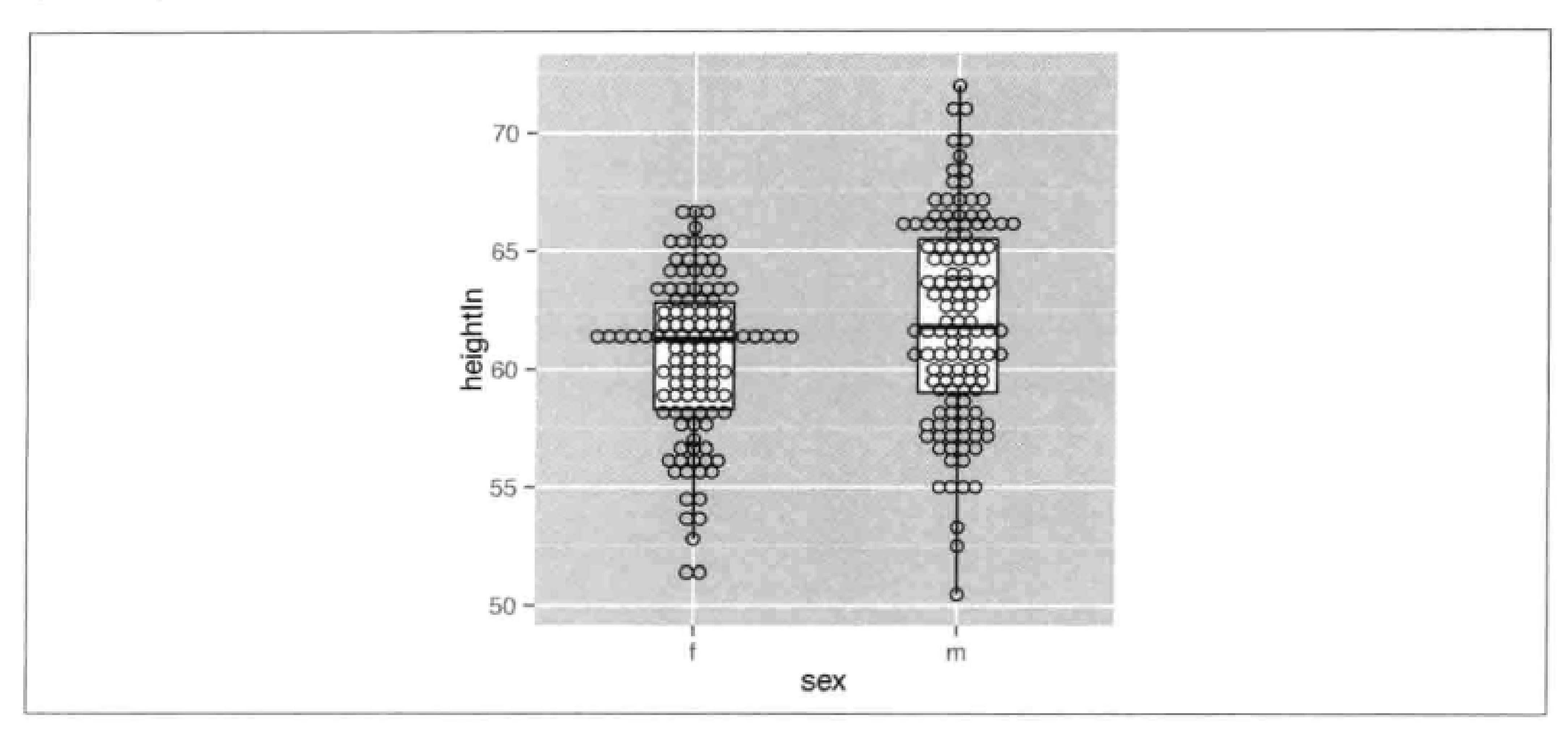


图 6-31　叠加在箱线图上的点图

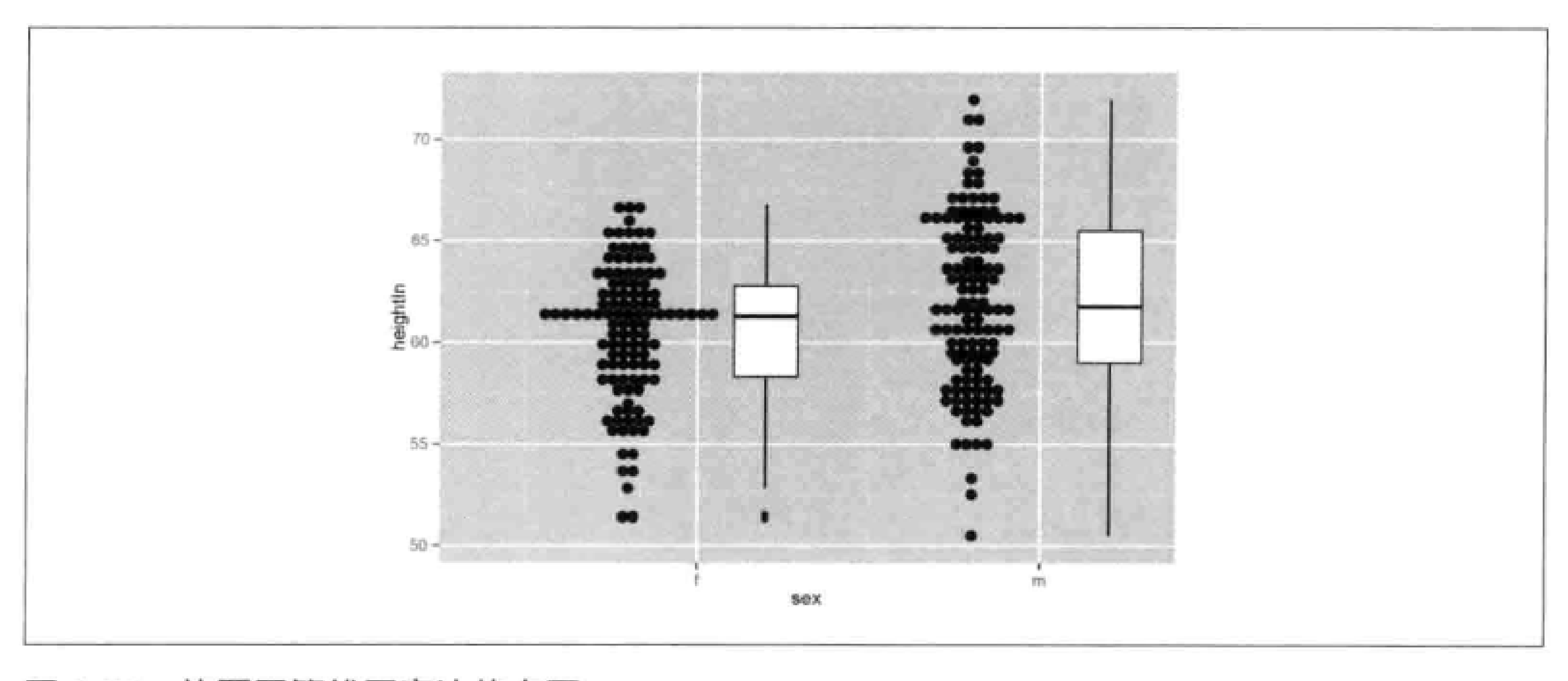


图 6-32　放置于箱线图旁边的点图

```
ggplot(heightweight, aes(x=sex, y=heightIn)) +
    geom_boxplot(aes(x=as.numeric(sex) + .2, group=sex), width=.25) +
    geom_dotplot(aes(x=as.numeric(sex) - .2, group=sex), binaxis="y",
                 binwidth=.5, stackdir="center") +
    scale_x_continuous(breaks=1:nlevels(heightweight$sex),
                       labels=levels(heightweight$sex))
```

6.12　绘制二维数据的密度图

问题

如何绘制二维（2D）数据的密度图？

方法

使用 stat_density2d() 函数。该函数会给出一个基于数据的二维核密度估计。首先，我们绘制数据点和密度等高线图（见图 6-33 左图）：

```
# 基础图
p <- ggplot(faithful, aes(x=eruptions, y=waiting))

p + geom_point() + stat_density2d()
```

也可以使用 ..level.. 将密度曲面的高度映射给等高线的颜色（见图 6-33 右图）：

```
# 将 height 映射到颜色的等高线
p + stat_density2d(aes(colour=..level..))
```

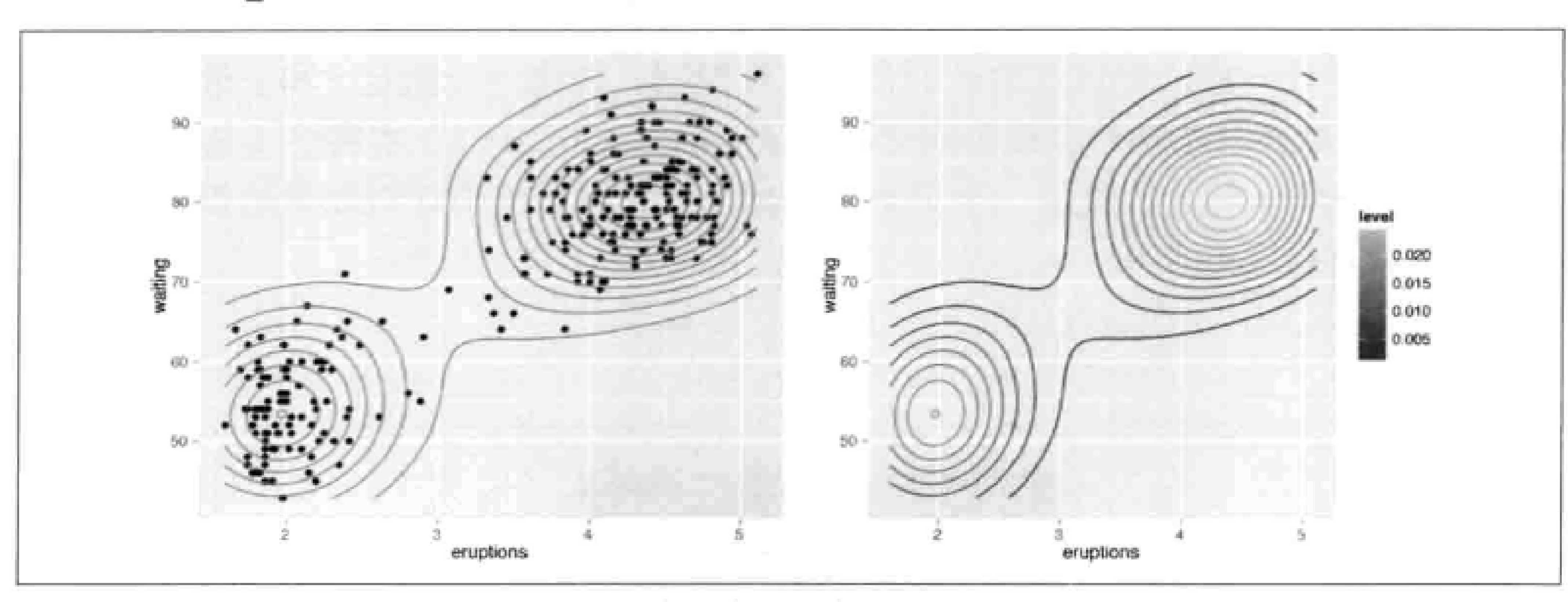


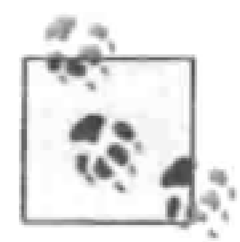
图 6-33　左图：数据点与密度等高线　右图：使用 **..level..** 将密度曲面的高度映射到颜色

讨论

二维核密度估计类似于 stat_density() 函数生成的一维核密度估计，不过，前者展示图形的方法有所不同。系统默认使用等高线，也可以使用瓦片图（tile）将密度估计映射给填充色或者瓦片图的透明度，如图 6-34 所示：

```
# 将密度估计映射给填充色
p + stat_density2d(aes(fill=..density..), geom="raster", contour=FALSE)

# 带数据点，并将密度估计映射给 alpha 的瓦片图
p + geom_point() +
    stat_density2d(aes(alpha=..density..), geom="tile", contour=FALSE)
```



前面的第一个例子中我们使用了 geom="raster"，而第二个例子中使用的是 geom="tile"。两者的主要区别在于栅格几何对象能够比瓦片更有效地进行渲染。理论上，两者应该看起来一样，但实际中两者常常不同。如果输出是 PDF 文件，则图形的外观会依赖于打开 PDF 的浏览器类型。在一些浏览器上，当使用瓦片时，瓦片中间可能会有模糊的线；而当使用栅格时，瓦片的边沿可能会显示模糊（尽管在本例中不存在这个问题）。

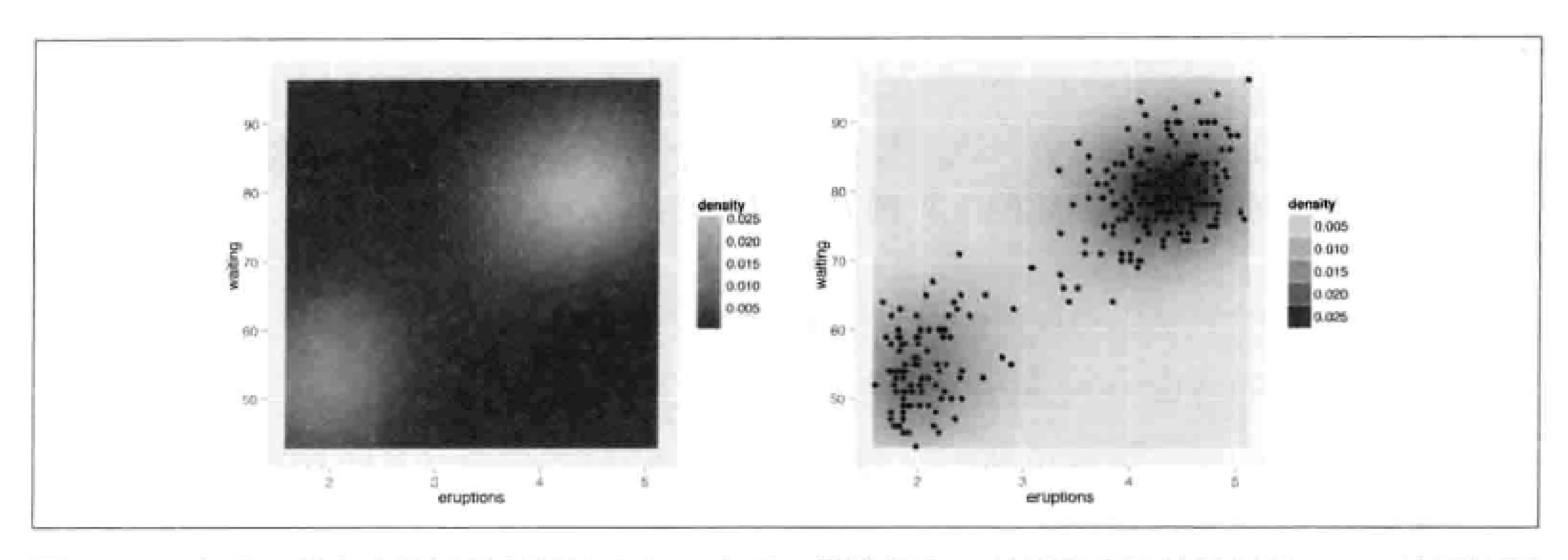


图 6-34　左图：将密度估计映射给填充色　右图：带数据点，并将密度估计映射给 alpha 的瓦片图

与一维密度估计一样，可以对估计的带宽进行控制。传递一个指定 x 和 y 带宽的向量到 h，这个参数会被传递给直接生成密度估计的函数 kde2d()。本例中（见图 6-35），我们将在 x 轴和 y 轴方向使用一个更小的带宽，以使得密度估计对数据的拟合程度更高（可能会有过度拟合）：

```
p + stat_density2d(aes(fill=..density..), geom="raster",
                   contour=FALSE, h=c(.5,5))
```

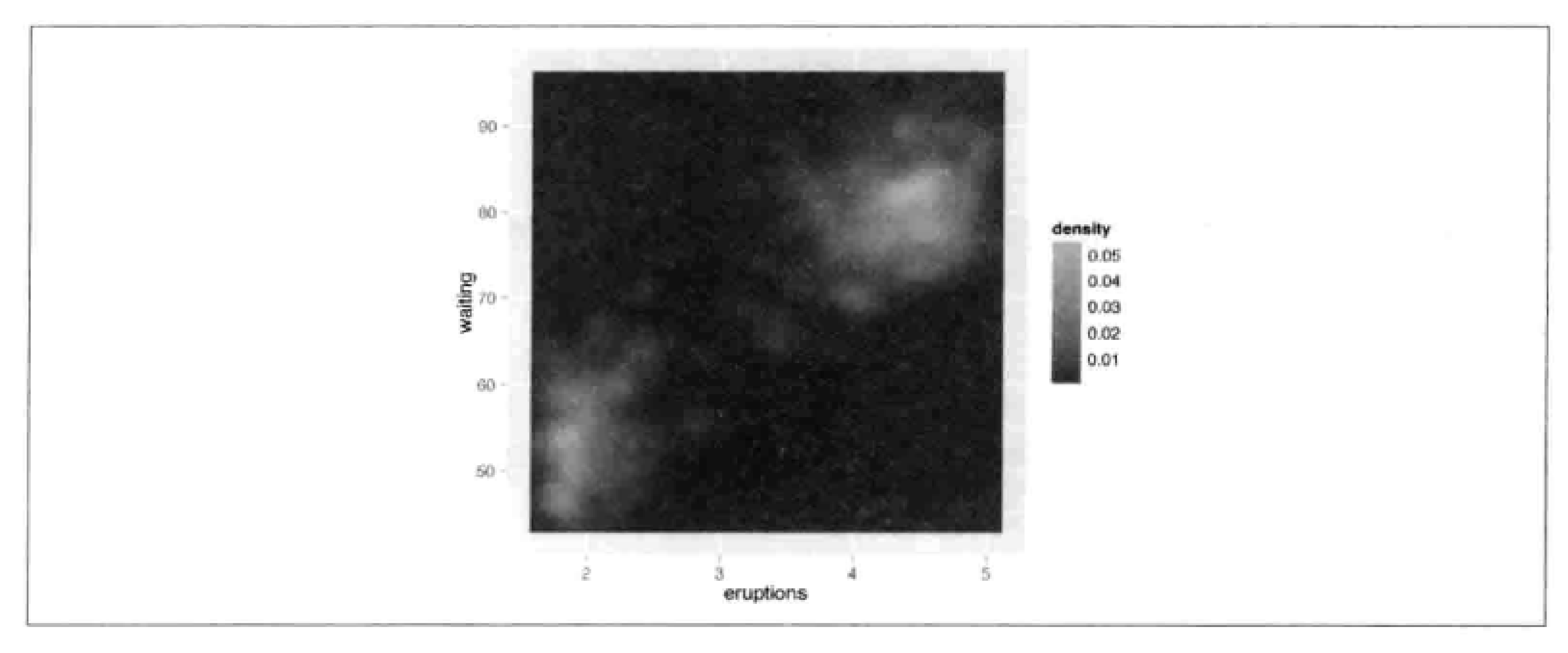


图 6-35　x 轴和 y 轴方向上带宽更小的密度图

另见

stat_density2d() 函数和 stat_bin2d() 函数的关系与它们各自的一维情形，即密度曲线和直方图之间的关系类似。密度曲线是在特定假设下对分布的估计，而分组可视化则是直接表示观测值。更多关于数据分组的内容参见 5.5 节。

如果想使用不同的调色板，参见 12.6 节。

stat_density2d() 可将选项传递到 kde2d() 函数；输入 ?kde2d 可以查看函数选项的信息。

第7章

注解

仅仅展示你的数据是不够的——还有许多各式各样的其他信息可以帮助看图者解读数据。除了坐标轴标签、刻度线和图例这些标准的保留元素，你还可以向图形添加独立的图形元素或文本元素。这些元素可用于增加额外的上下文信息、高亮图形的某个区域，或是补充一些关于数据的描述性文本。

7.1 添加文本注解

问题

如何向图形添加一条文本注解？

方法

使用 annotate() 和一个文本类几何对象（见图 7-1）：

```
p <- ggplot(faithful, aes(x=eruptions, y=waiting)) + geom_point()

p + annotate("text", x=3, y=48, label="Group 1") +
    annotate("text", x=4.5, y=66, label="Group 2")
```

讨论

函数 annotate() 可以用于添加任意类型的几何对象。在本例中，我们使用的几何对象是 geom="text"。

我们也可指定其他文本属性，如图 7-2 所示：

```
p + annotate("text", x=3, y=48, label="Group 1", family="serif",
             fontface="italic", colour="darkred", size=3) +
   annotate("text", x=4.5, y=66, label="Group 2", family="serif",
             fontface="italic", colour="darkred", size=3)
```

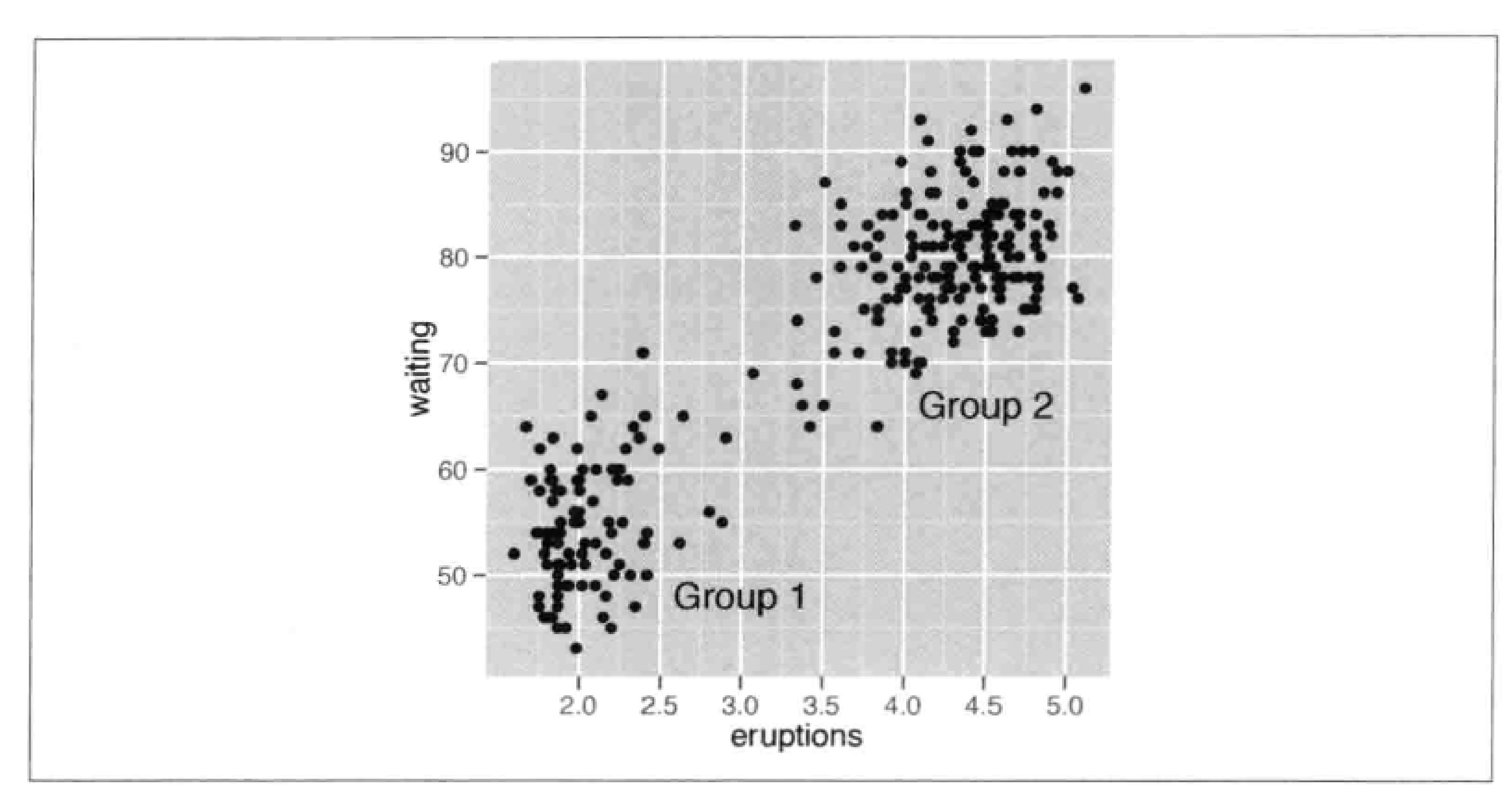


图 7-1　文本注解

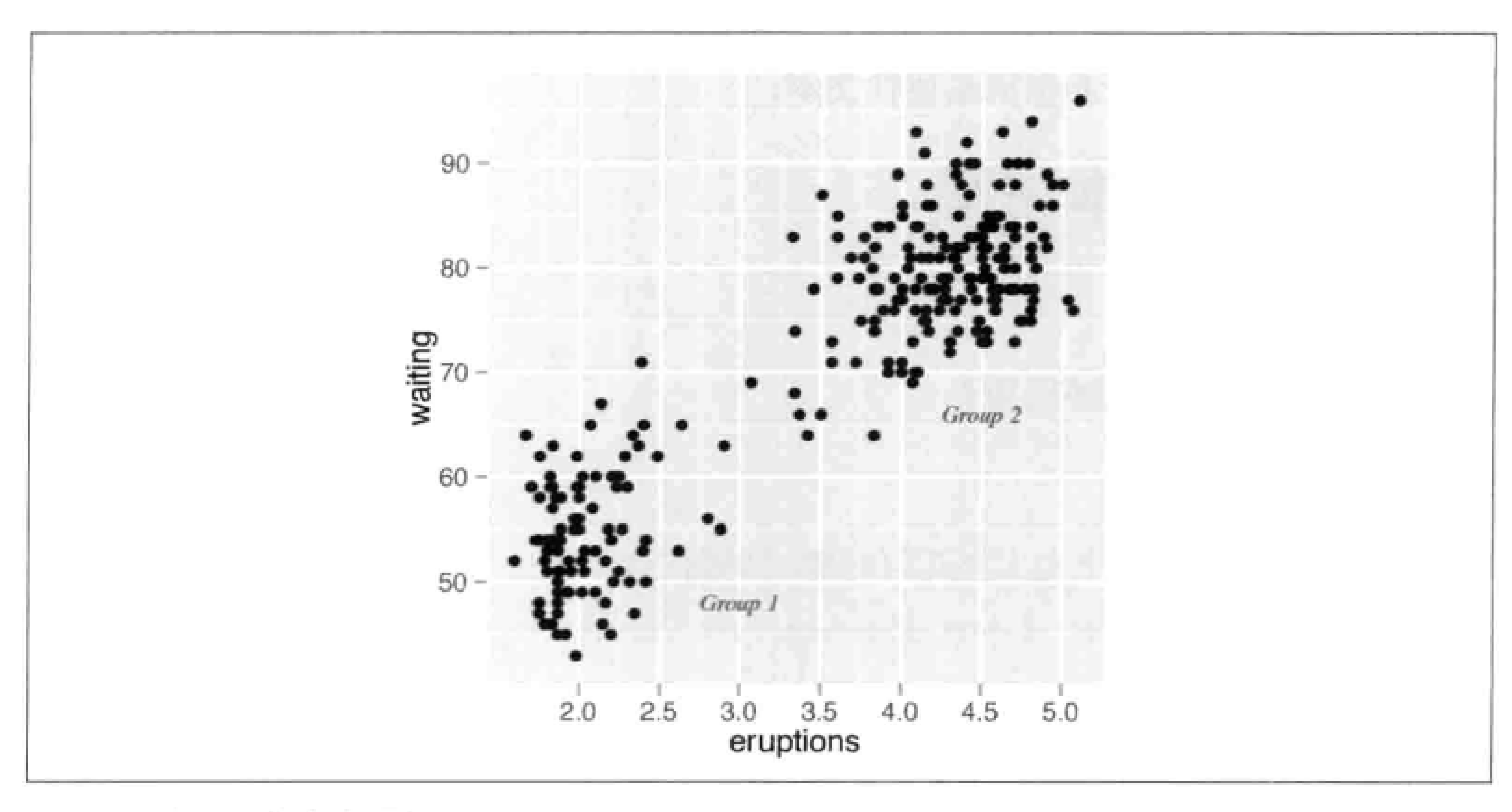


图 7-2　修改后的文本属性

当你希望添加独立的文本对象时，千万不要使用 geom_text()。annotate(geom="text") 会向图形添加一个单独的文本对象，而 geom_text() 却会根据数据创建许多的文本对象，如 5.11 节讨论过的那样。

如果使用 geom_text()，文本会在相同的位置被严重地遮盖，每个数据点各重绘了一次：

```
p + annotate("text", x=3, y=48, label="Group 1", alpha=.1) +     # 正常
    geom_text(x=4.5, y=66, label="Group 2", alpha=.1)            # 遮盖绘制
```

在图 7-3 中，每个文本标签都是 90% 透明的，这样就很清楚地显示出了哪一个被遮盖绘制了。在输出为点阵格式时，遮盖绘制问题可能会导致边缘走样（有锯齿）。

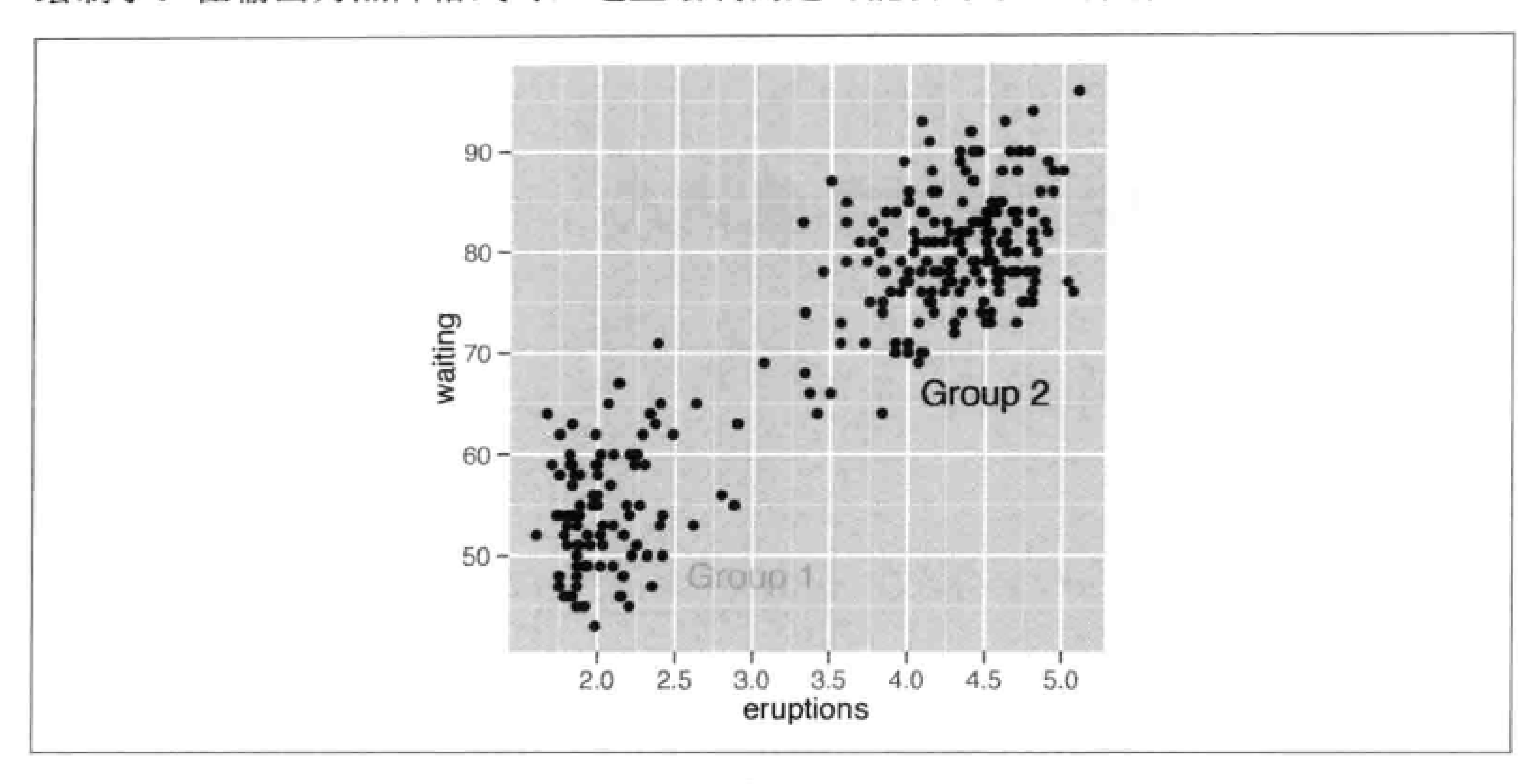


图 7-3　其中一个标签的遮盖绘制现象——事实上两个标签都应为 90% 透明

如果坐标轴是连续型的，你可以使用特殊值 Inf 和 -Inf 在绘图区域的边缘放置文本注解，如图 7-4 所示。同时，也需要使用 hjust 和 vjust 来调整文本相对于边角的位置——如果你让它们留在默认值的位置上，这些文本就会居中于边界线之上。要将文本定位到理想的位置，可能需要进行一些尝试：

```
p + annotate("text", x=-Inf, y=Inf, label="Upper left", hjust=-.2, vjust=2) +
    annotate("text", x=mean(range(faithful$eruptions)), y=-Inf, vjust=-0.4,
             label="Bottom middle")
```

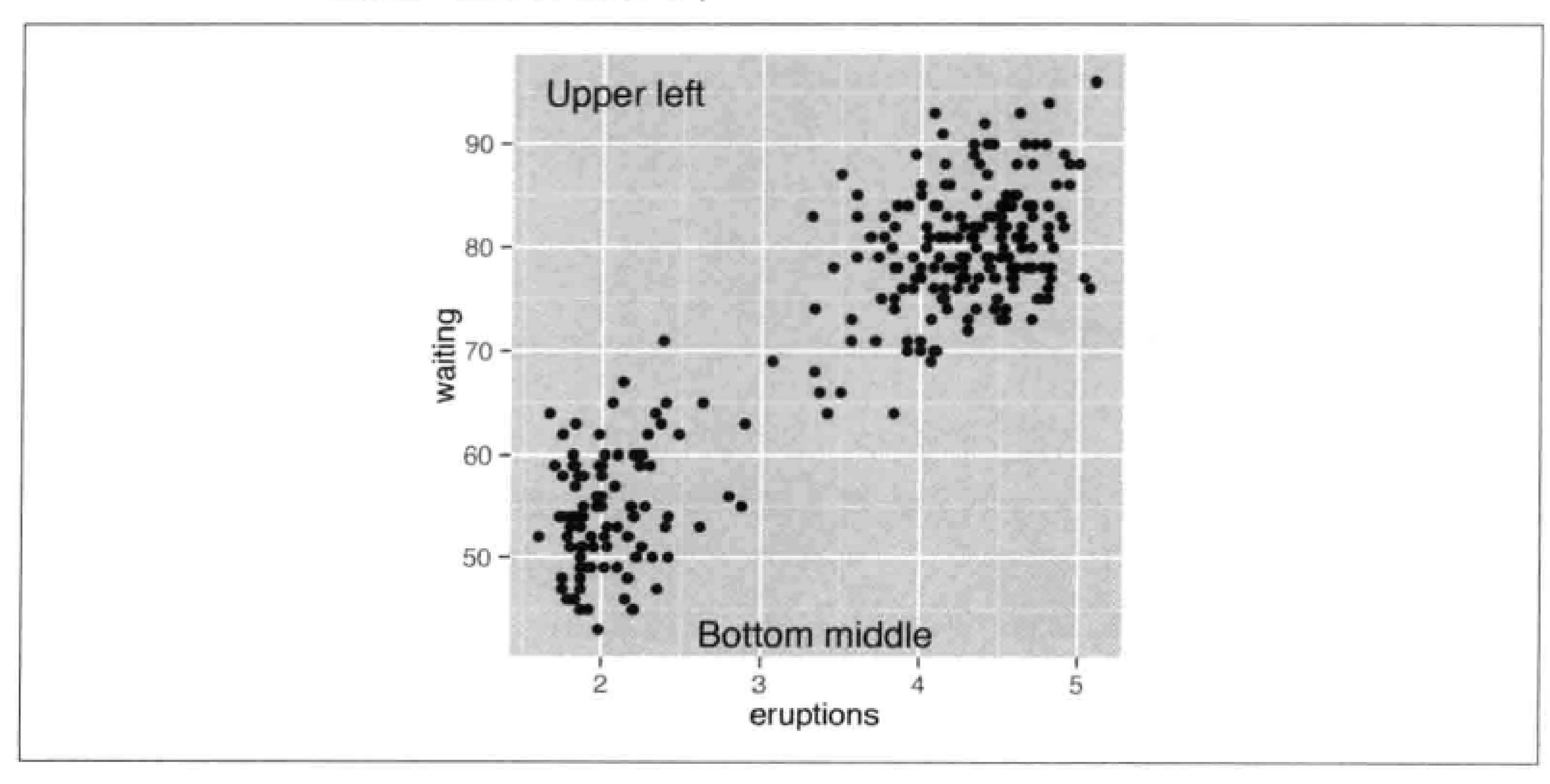


图 7-4　置于绘图区域边缘的文本

另见

参见 5.11 节以绘制带有文本的散点图。

关于控制文本外观的更多方法，参见 9.2 节。

7.2 在注解中使用数学表达式

问题

如何添加一条含数学符号的文本注解？

方法

使用 annotate(geom="text") 并设置 parse=TRUE（见图 7-5）：

```
# 一条正态曲线
p <- ggplot(data.frame(x=c(-3,3)), aes(x=x)) + stat_function(fun = dnorm)

p + annotate("text", x=2, y=0.3, parse=TRUE,
            label="frac(1, sqrt(2 * pi)) * e ^ {-x^2 / 2}")
```

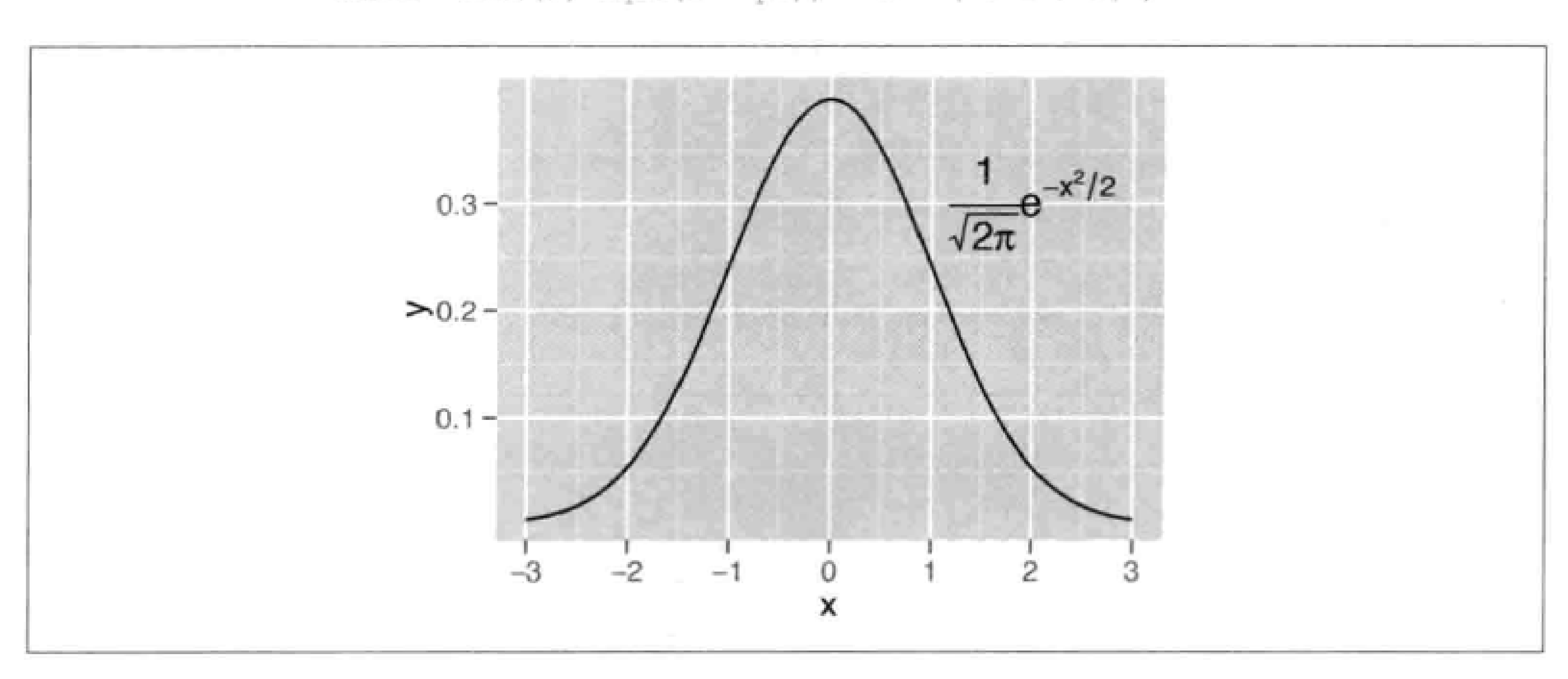


图 7-5 含数学表达式的注解

讨论

在 ggplot2 中使用 parse=TRUE 和文本类几何对象创建的数学表达式，和那些在 R 基础图形中利用 plotmath 和 expression 创建的数学表达式有着类似的格式，唯一的区别是，前者以字符串的形式存储，而后者是表达式对象。

要将常规文本融入表达式中，只需在双引号内使用单引号（或者反过来）标出纯文本的部分即可。通过内部引号闭合的每一块文本都将被作为数学表达式中的一个变量对待。切记，在 R 的数学表达式语法中，你不能简单地把一个变量直接放到另一个变量

旁边却不在中间加上任何记号。如图 7-6 所示，要显示两个相邻的变量，需要在它们之间放置一个 * 操作符；在显示图形时，它会被当作一个不可见的乘号对待（要显示一个可见的乘号，需要使用 %*%）：

```
p + annotate("text", x=0, y=0.05, parse=TRUE, size=4,
             label="'Function: ' * y==frac(1, sqrt(2*pi)) * e^(-x^2/2)")
```

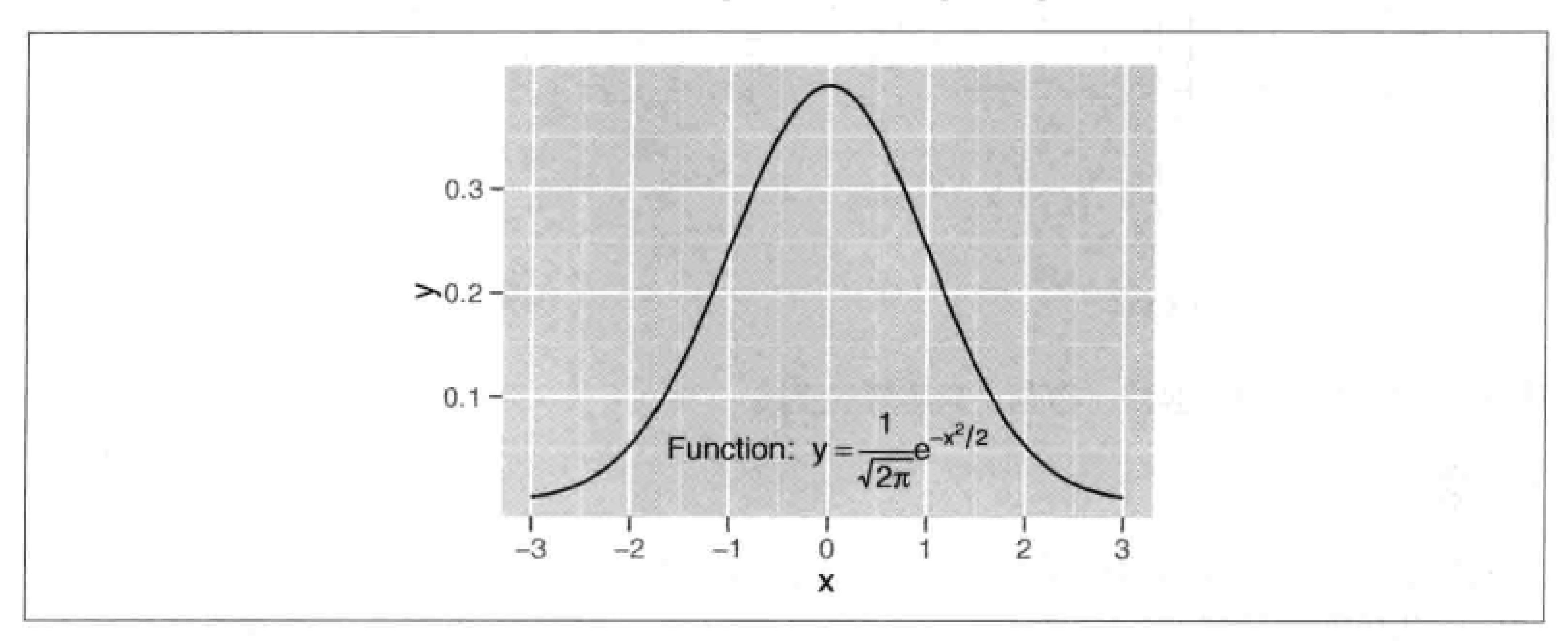


图 7-6 含常规文本的数学表达式

另见

更多数学表达式的示例可见 ?plotmath，数学表达式的图示参见 ?demo(plotmath)。

参见 5.9 节以向图形添加回归系数。

要在数学表达式中使用其他字体，参见 14.6 节。

7.3 添加直线

问题

如何向图形添加直线？

方法

对于横线和竖线，使用 geom_hline() 和 geom_vline() 即可。对于有角度的直线，则使用 geom_abline()（见图 7-7）。对于下例，我们将使用 heightweight 数据集：

```
library(gcookbook)  # 为了使用数据集

p <- ggplot(heightweight, aes(x=ageYear, y=heightIn, colour=sex)) + geom_point()

# 添加横线和竖线
p + geom_hline(yintercept=60) + geom_vline(xintercept=14)
```

```
# 添加有角度的直线
p + geom_abline(intercept=37.4, slope=1.75)
```

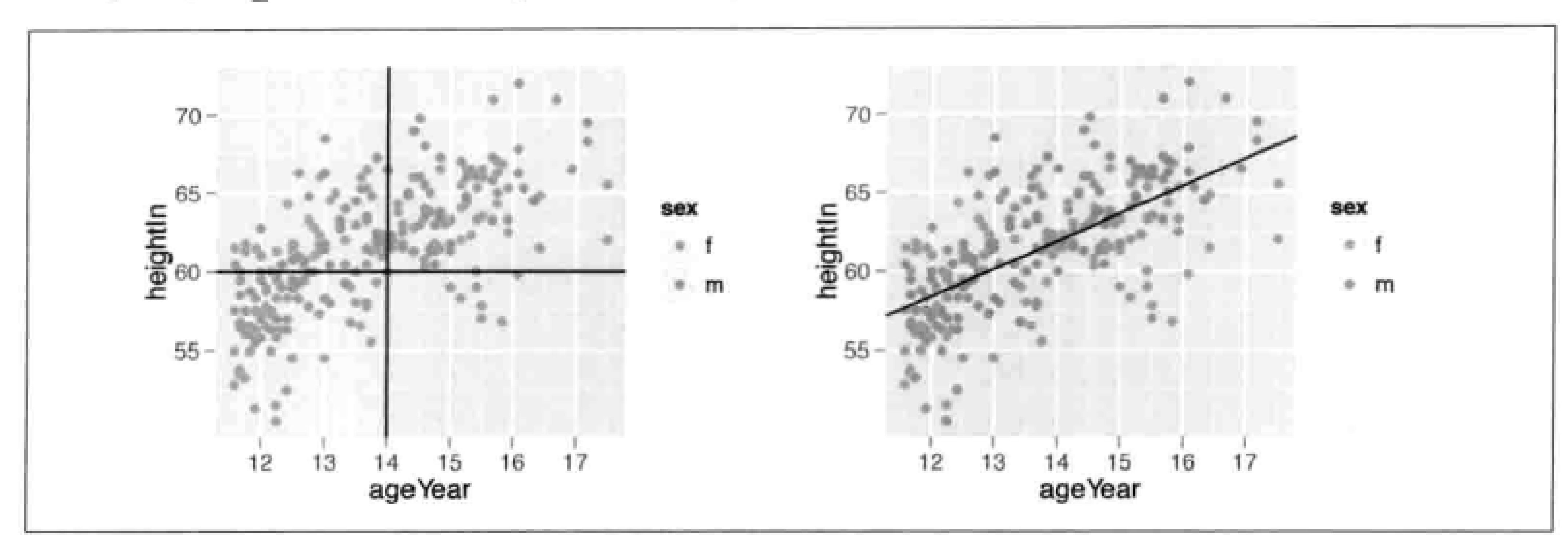


图 7-7　左图：横线和竖线　右图：有角度的直线

讨论

上例演示了手动设置直线位置的方法，效果是每添加一个几何对象绘制一条线。我们也可以将值从数据映射到 xintercept、yintercept 等之上，甚至是绘制另一个数据框中的值。

我们将在这里计算男性和女性的平均身高，并将它们存储到一个数据框 hw_means 中。然后为每个均值绘制一条水平线，并手工设定 linetype 和 size（见图 7-8）：

```
library(plyr)  # 为了使用 ddply() 函数
hw_means <- ddply(heightweight, "sex", summarise, heightIn=mean(heightIn))
hw_means

sex heightIn
  f 60.52613
  m 62.06000

p + geom_hline(aes(yintercept=heightIn, colour=sex), data=hw_means,
               linetype="dashed", size=1)
```

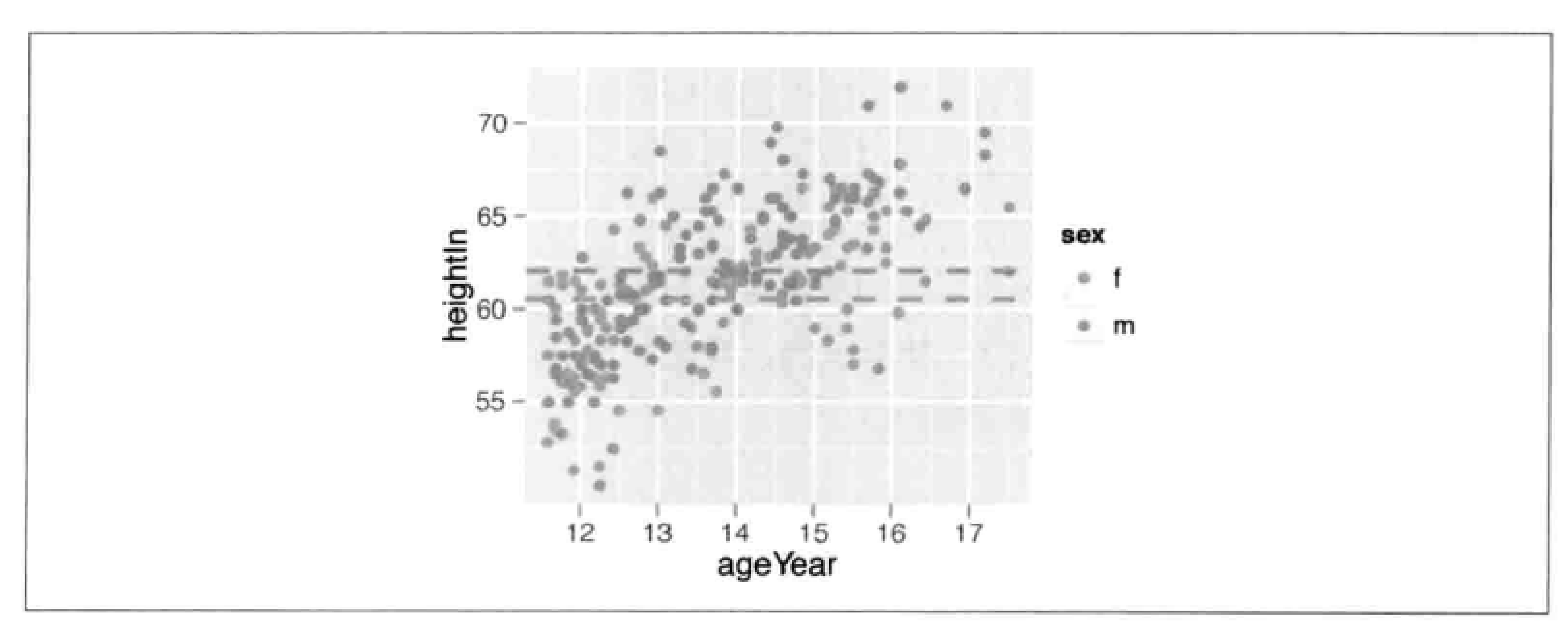


图 7-8　在每组均值处绘制的多条直线

如果某个坐标轴是离散型而不是连续型的，不能以字符串的形式直接指定截距——必须仍以数字的形式指定它们。假设此坐标轴表示一个因子，那么第一个水平为数值 1，第二个水平为数值 2，依次类推。可以像下面这样手工指定数值型的截距，或者使用 `which(levels(...))` 计算所需数值（见图 7-9）：

```
pg <- ggplot(PlantGrowth, aes(x=group, y=weight)) + geom_point()

pg + geom_vline(xintercept = 2)

pg + geom_vline(xintercept = which(levels(PlantGrowth$group)=="ctrl"))
```

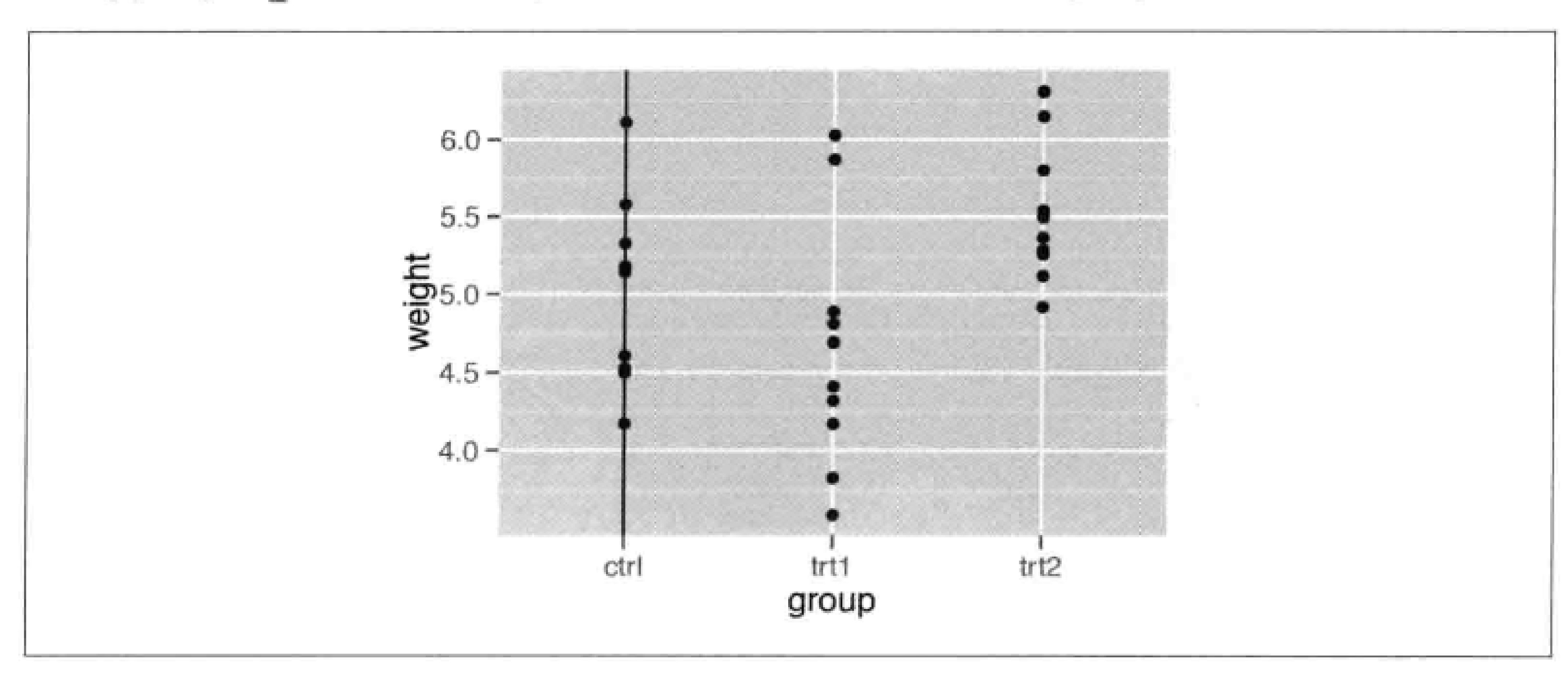
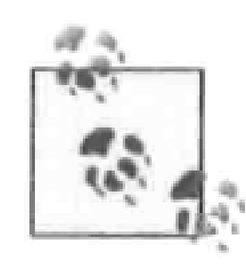


图 7-9　离散型坐标轴上的直线



你可能已经注意到，添加直线与添加其他的注解有所不同。我们使用了 `geom_hline()` 和其他相关函数，却并未使用 `annotate()` 函数。这是因为 ggplot2 的早期版本中并没有 `annotate()` 函数。直线几何对象过去被编码用于处理添加单条直线的特殊情况，修改它将会破坏向下兼容性。在未来某个版本的 ggplot2 中，这一点将会有所改变，`annotate()` 将可以配合直线几何对象正常工作。

另见

要添加回归曲线，参见 5.6 节和 5.7 节。

线条经常用于显示数据的概要信息。参见 15.17 节以了解更多按照组计算数据概要的方法。

7.4　添加线段和箭头

问题

如何向图形添加线段或箭头？

方法

使用 `annotate("segment")`。在本例中，我们将使用 `climate` 数据集中数据来源为 Berkeley 的子集（见图 7-10）：

```
library(gcookbook) # 为了使用数据集

p <- ggplot(subset(climate, Source=="Berkeley"), aes(x=Year, y=Anomaly10y)) +
       geom_line()

p + annotate("segment", x=1950, xend=1980, y=-.25, yend=-.25)
```

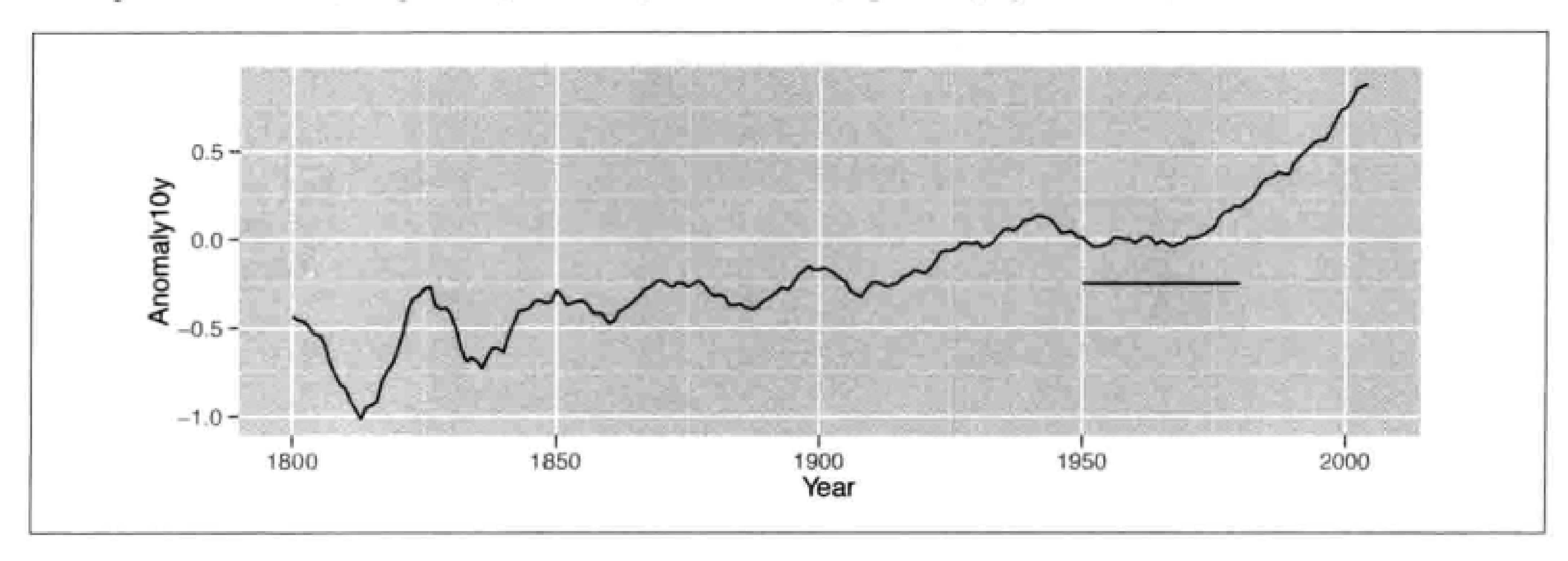


图 7-10 线段注解

讨论

可以使用 grid 包中的 `arrow()` 函数向线段两端添加箭头或平头。在本例中，我们将一并演示（见图 7-11）：

```
library(grid)
p + annotate("segment", x=1850, xend=1820, y=-.8, yend=-.95, colour="blue",
             size=2, arrow=arrow()) +
    annotate("segment", x=1950, xend=1980, y=-.25, yend=-.25,
             arrow=arrow(ends="both", angle=90, length=unit(.2,"cm")))
```

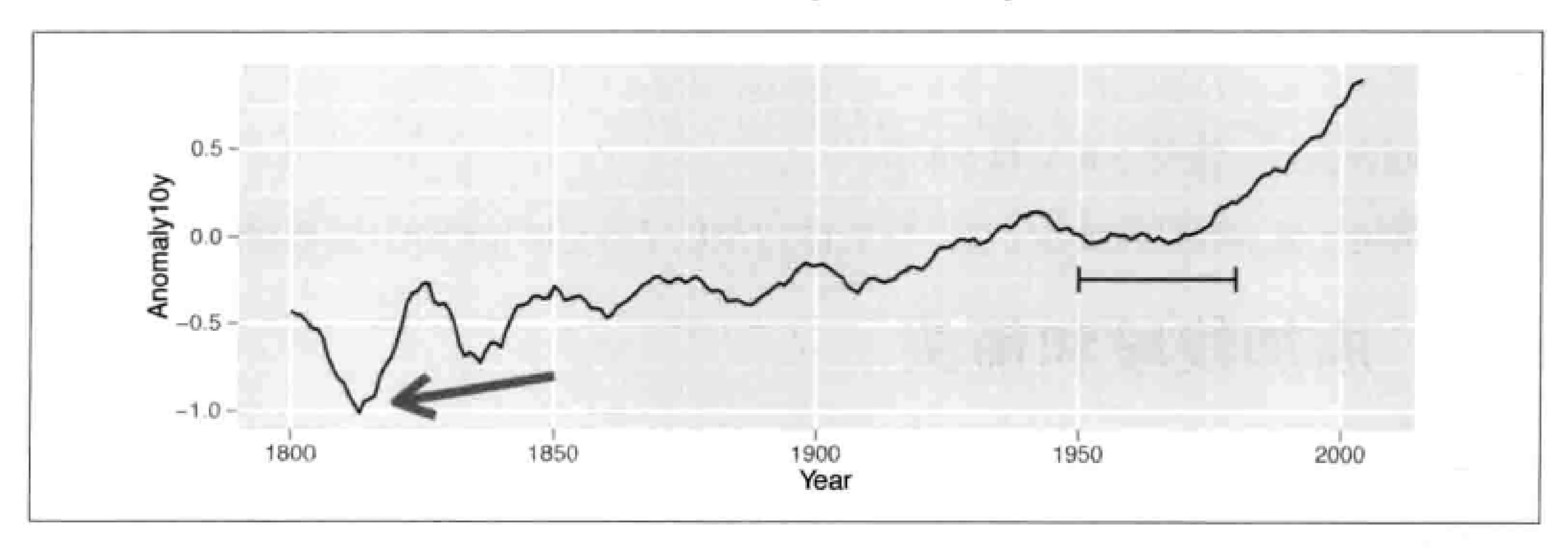


图 7-11 带有箭头的线段

箭头线的默认角度（angle）为 30 度，默认长度（length）为 0.2 英寸（0.508 厘米）。如果一个或多个坐标轴是离散型的，则 x 和 y 的位置即由拥有坐标值 1, 2, 3 等的类别项表示。

另见

要了解关于绘制箭头所需参数的更多信息，请载入 grid 包后查看 ?arrow。

7.5 添加矩形阴影

问题

如何添加一个阴影区域？

方法

使用 annotate("rect")（见图 7-12）：

```
library(gcookbook) # 为了使用数据集

p <- ggplot(subset(climate, Source=="Berkeley"), aes(x=Year, y=Anomaly10y)) +
    geom_line()

p + annotate("rect", xmin=1950, xmax=1980, ymin=-1, ymax=1, alpha=.1,
             fill="blue")
```

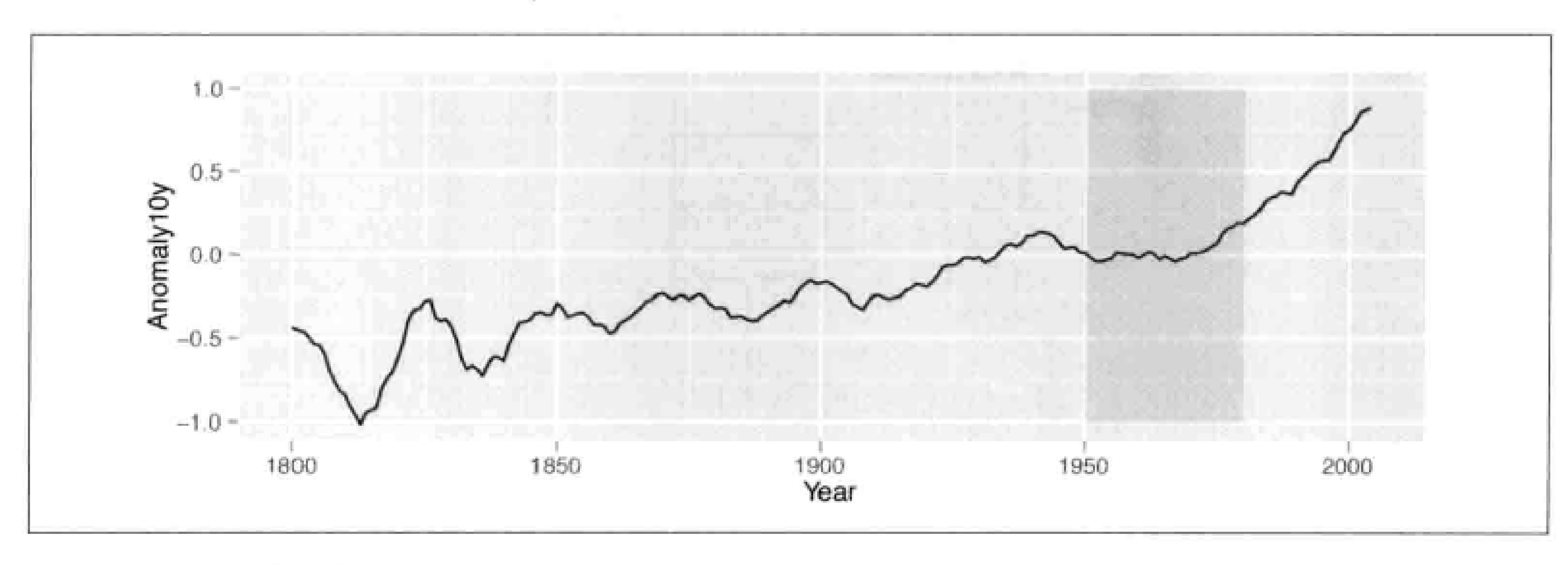


图 7-12 一个矩形阴影

讨论

每一个图层都是按照添加到 ggplot 对象的先后顺序绘制的，所以上例中的矩形被绘制在了曲线上方。在这个例子中这不成问题，但是如果你想把曲线放在矩形上方，则要先添加矩形，再添加曲线。

只要传递了合适的参数，任意几何对象都可以配合 annotate() 使用。在本例中，geom_rect() 所需的参数是 x 和 y 的最大值与最小值。

7.6 高亮某一元素

问题

如何修改某一元素的颜色以使其突出显示？

方法

要高亮一个或多个元素，需要在数据中创建一个新列并将其映射为颜色。在本例中，我们将创建一个新列 hl，并根据 group 的值来设定它的值：

```
pg <- PlantGrowth                          # 复制一份 PlantGrowth 数据
pg$hl <- "no"                              # 设定所有值为 "no"
pg$hl[pg$group=="trt2"] <- "yes"           # 如果 group 值为 "trt2"，设定为 "yes"
```

接下来使用手工指定的颜色画图，且不加图例（见图 7-13）：

```
ggplot(pg, aes(x=group, y=weight, fill=hl)) + geom_boxplot() +
    scale_fill_manual(values=c("grey85", "#FFDDCC"), guide=FALSE)
```

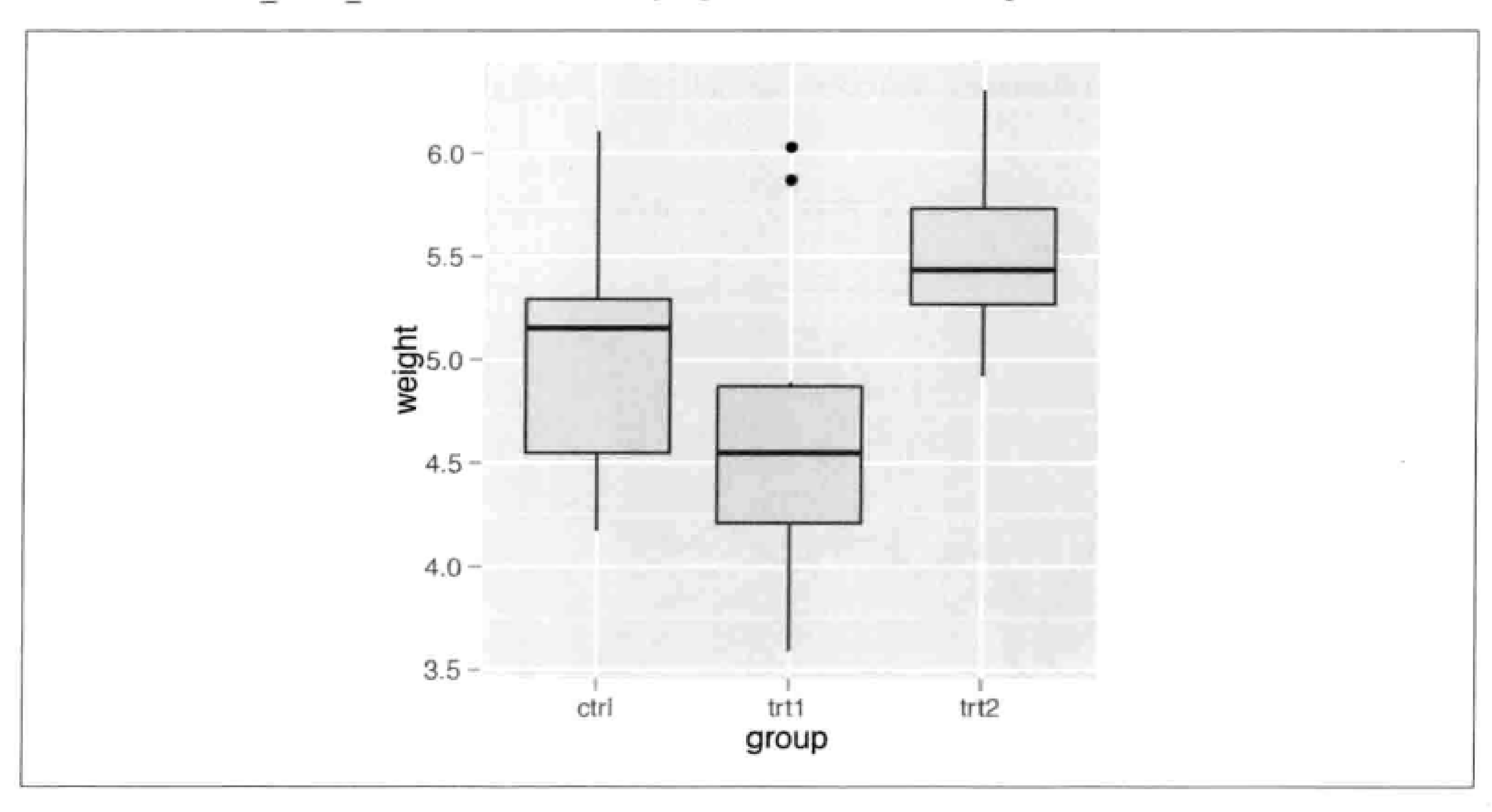


图 7-13　高亮一个元素

讨论

如果你有像本例一样少量的元素要高亮，那么可以使用原始变量并为每个水平指定颜色，而不必创建一个新列。举例来说，下列代码将使用 PlantGrowth 数据中的 group 列，并手工为三个水平逐个设定颜色。结果看起来与前述代码相同：

```
ggplot(PlantGrowth, aes(x=group, y=weight, fill=group)) + geom_boxplot() +
    scale_fill_manual(values=c("grey85", "grey85", "#FFDDCC"), guide=FALSE)
```

另见

参见第 12 章以了解关于如何指定颜色的更多信息。

关于移除图例的更多说明，参见 10.1 节。

7.7 添加误差线

问题

如何向图形添加误差线？

方法

使用 geom_errorbar 并将变量映射到 ymin 和 ymax 的值即可。对于条形图和折线图，添加误差线的方法相同，如图 7-14 所示（尽管条形图与折线图 y 的默认范围有所不同）：

```
library(gcookbook)  # 为了使用数据集
# 为本例抽取 cabbage_exp 数据的一个子集
ce <- subset(cabbage_exp, Cultivar == "c39")

# 为条形图添加误差线
ggplot(ce, aes(x=Date, y=Weight)) +
    geom_bar(fill="white", colour="black") +
    geom_errorbar(aes(ymin=Weight-se, ymax=Weight+se), width=.2)

# 为折线图添加误差线
ggplot(ce, aes(x=Date, y=Weight)) +
    geom_line(aes(group=1)) +
    geom_point(size=4) +
    geom_errorbar(aes(ymin=Weight-se, ymax=Weight+se), width=.2)
```

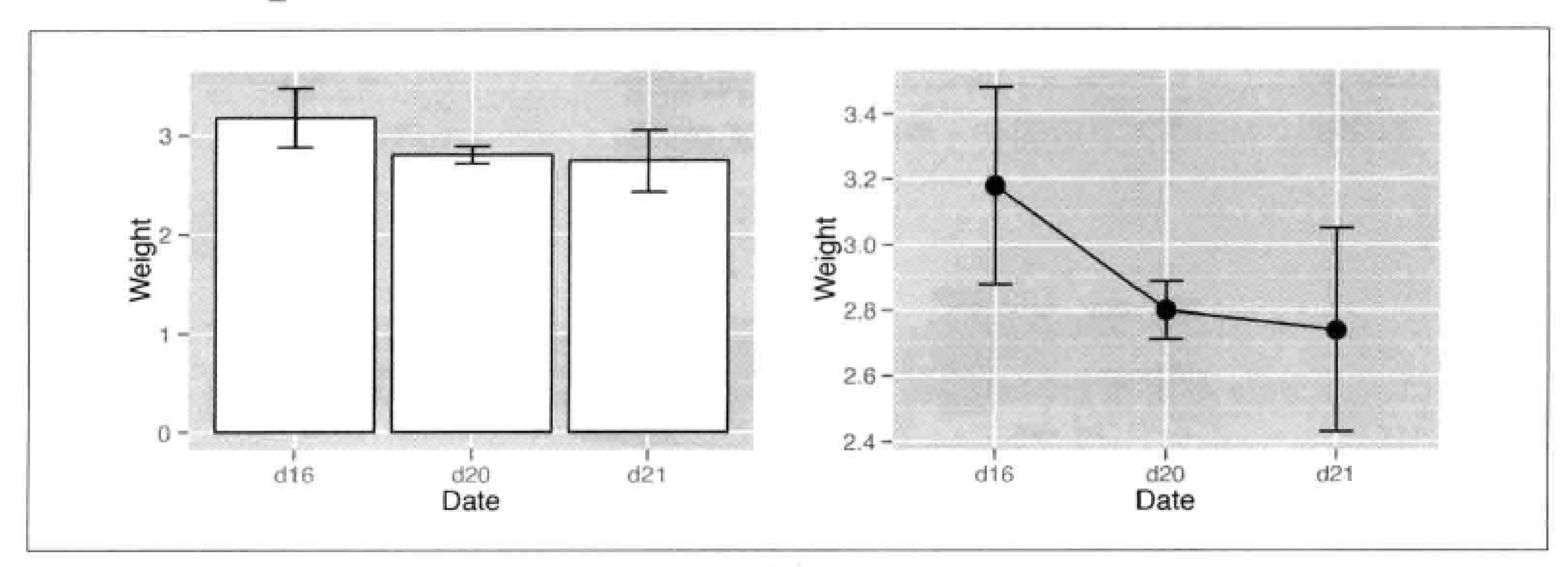


图 7-14　左图：条形图上的误差线　右图：折线图上的误差线

讨论

在本例中，数据已经包含了均值的标准误差（se），我们将利用它来绘制误差线（数

据中也包含了标准差（sd）的数据，不过我们在这里用不着它）：

```
ce

Cultivar Date Weight        sd  n         se
     c39  d16   3.18 0.9566144 10 0.30250803
     c39  d20   2.80 0.2788867 10 0.08819171
     c39  d21   2.74 0.9834181 10 0.31098410
```

为了获得 ymax 和 ymin 的值，我们取 y 值变量 Weight 加上和减去了 se。

同样，我们使用 width=.2 指定了误差线的末端宽度。要找到看起来理想的值，最好进行一些试探。如果你不设置这个宽度，则误差线将会非常宽，横跨 *x* 轴上各项的全部空间。

对于一幅分组的条形图，各误差线也必须被并列（dodged）；否则，它们会有完全相同的 *x* 坐标，无法与条形对齐（参见 3.2 节以了解关于分组条形图和并列的更多信息）。

这次我们将利用完整的 cabbage_exp 数据集：

```
cabbage_exp

Cultivar Date Weight        sd  n         se
     c39  d16   3.18 0.9566144 10 0.30250803
     c39  d20   2.80 0.2788867 10 0.08819171
     c39  d21   2.74 0.9834181 10 0.31098410
     c52  d16   2.26 0.4452215 10 0.14079141
     c52  d20   3.11 0.7908505 10 0.25008887
     c52  d21   1.47 0.2110819 10 0.06674995
```

geom_bar() 的默认并列宽度为 0.9，你必须让误差线的并列宽度与此相同。如果不指定并列宽度，则默认按误差线的宽度并列，而此宽度通常会小于条形的宽度（见图 7-15）：

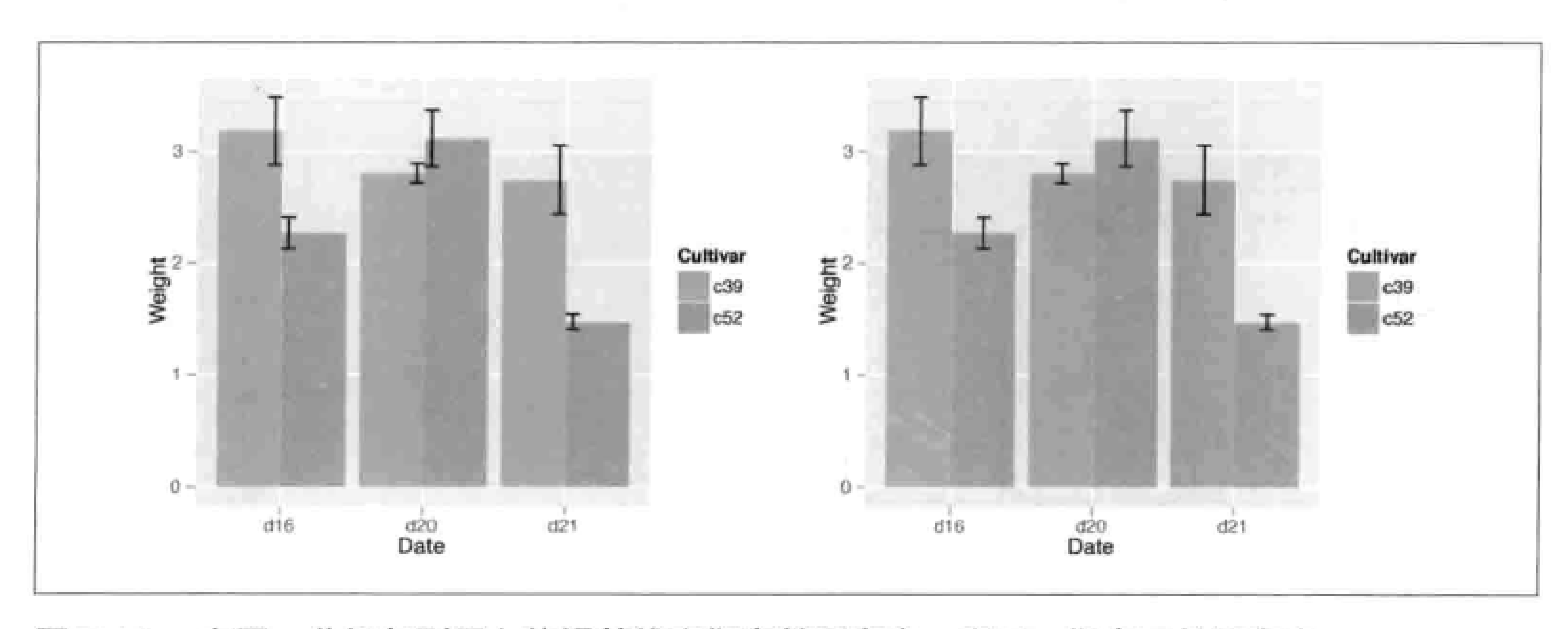


图 7-15　左图：分组条形图上的误差线未指定并列宽度　右图：指定了并列宽度

```
# 反例：未指定并列宽度
ggplot(cabbage_exp, aes(x=Date, y=Weight, fill=Cultivar)) +
    geom_bar(position="dodge") +
    geom_errorbar(aes(ymin=Weight-se, ymax=Weight+se),
```

```
                position="dodge", width=.2)

# 正例：设定并列宽度与条形的相同 (0.9)
ggplot(cabbage_exp, aes(x=Date, y=Weight, fill=Cultivar)) +
    geom_bar(position="dodge") +
    geom_errorbar(aes(ymin=Weight-se, ymax=Weight+se),
                position=position_dodge(0.9), width=.2)
```



注意我们在第一个版本中使用了 position="dodge"，即 position=position_dodge() 的简写。但如果要传递一个特定值，我们必须把它完整地拼出，如 position_dodge(0.9)。

对于折线图来说，如果误差线的颜色与线和点的颜色不同，则应先绘制误差线，这样它们就会位于点和线的下层了。否则，误差线将被绘制在点和线的上层，看起来会不太对。

另外，你应当同时并列所有的几何元素，这样它们就会同误差线对齐，如图 7-16 所示：

```
pd <- position_dodge(.3)  # 保存并列参数，因为我们要重复使用它

ggplot(cabbage_exp, aes(x=Date, y=Weight, colour=Cultivar, group=Cultivar)) +
    geom_errorbar(aes(ymin=Weight-se, ymax=Weight+se),
                 width=.2, size=0.25, colour="black", position=pd) +
     geom_line(position=pd) +
     geom_point(position=pd, size=2.5)

# 使用 size=0.25 绘制更细的误差线线条，使用 size=2.5 绘制更大的点
```

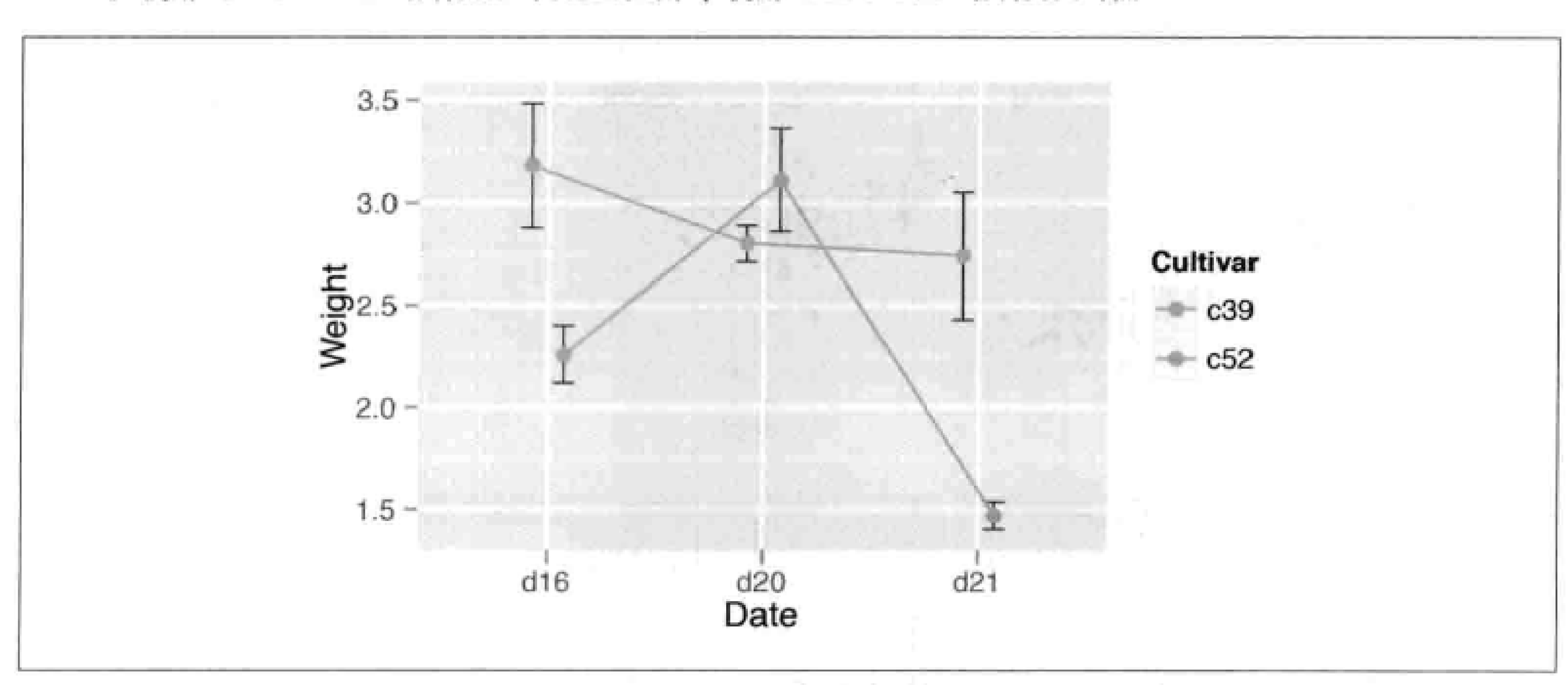


图 7-16　折线图上的误差线，已被并列所以不会相互重叠

注意，我们设置了 colour="black" 以使误差线为黑色；否则它们将继承上方的 colour 值。我们也通过将 Cultivar 映射到 group 的方式来确保它被作为分组变量使用。

当一个离散型变量被映射到一个如 colour 或 fill 的图形属性时（如条形的例子），此变量就会被用于对数据进行分组。但是通过设定误差线的 colour 属性，我们将使

这个针对 colour 的变量并没用于分组，所以需要一些其他的途径来通知 ggplot() 在每个 *x* 位置的这两组数据属于不同的组，这样它们就可以被正确地并列了。

另见

参见 3.2 节以了解更多关于创建分组条形图的信息，参见 4.3 节以了解如何创建含多条线的折线图。

参见 15.18 节以计算均值、标准差、标准误差和置信域等统计汇总。

参见 4.9 节以在数据沿 *x* 轴有更高密度时添加一个置信域。

7.8 向独立分面添加注解

问题

如何向图形中的各个分面添加注解？

方法

使用分面变量创建一个新的数据框，并设定每个分面要绘制的值。然后配合新数据框使用 geom_text()（见图 7-17）：

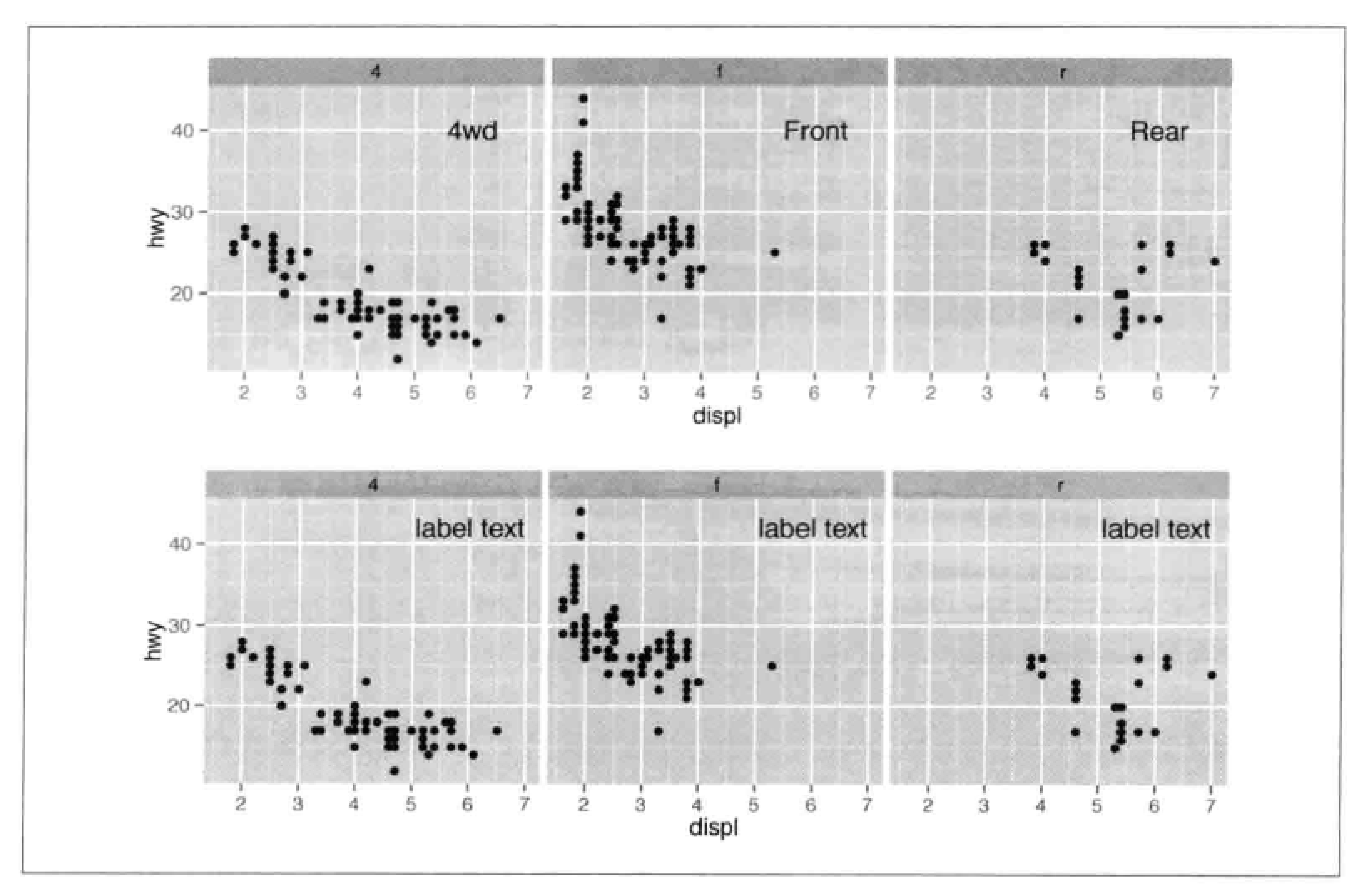


图 7-17 上图：每个分面的注解不同 下图：每个分面的注解相同

```
# 基本图形
p <- ggplot(mpg, aes(x=displ, y=hwy)) + geom_point() + facet_grid(. ~ drv)

# 存有每个分面所需标签的数据框
f_labels <- data.frame(drv = c("4", "f", "r"), label = c("4wd", "Front", "Rear"))

p + geom_text(x=6, y=40, aes(label=label), data=f_labels)

# 如果你使用 annotate()，标签将在所有分面上出现
p + annotate("text", x=6, y=42, label="label text")
```

讨论

这种方法可以用于显示每个分面中关于数据的信息，如图 7-18 所示。举例来说，我们可以展示每个分面的线性回归曲线、每条曲线的回归公式，以及 r^2 的值。要完成这件任务，我们将编写一个函数，它可以输入一个数据框并返回另外一个数据框，其中包含一个含有回归公式的字符串和一个含 r^2 值的字符串。然后我们使用 ddply() 将这个函数应用到每一组数据上：

```
# 此函数返回一个数据框，其中的字符串
# 表示回归公式和 r^2 值
# 这些字符串将被认为是 R 中的数学表达式
lm_labels <- function(dat) {
  mod <- lm(hwy ~ displ, data=dat)
  formula <- sprintf("italic(y) == %.2f %+.2f * italic(x)",
                     round(coef(mod)[1], 2), round(coef(mod)[2], 2))

  r <- cor(dat$displ, dat$hwy)
  r2 <- sprintf("italic(R^2) == %.2f", r^2)
  data.frame(formula=formula, r2=r2, stringsAsFactors=FALSE)
}

library(plyr)  # 为了使用 ddply() 函数
labels <- ddply(mpg, "drv", lm_labels)
labels

 drv                             formula                    r2
   4 italic(y) == 30.68 -2.88 * italic(x) italic(R^2) == 0.65
   f italic(y) == 37.38 -3.60 * italic(x) italic(R^2) == 0.36
   r italic(y) == 25.78 -0.92 * italic(x) italic(R^2) == 0.04

# 绘制公式和 R^2 值
p + geom_smooth(method=lm, se=FALSE) +
    geom_text(x=3, y=40, aes(label=formula), data=labels, parse=TRUE, hjust=0) +
    geom_text(x=3, y=35, aes(label=r2), data=labels, parse=TRUE, hjust=0)
```

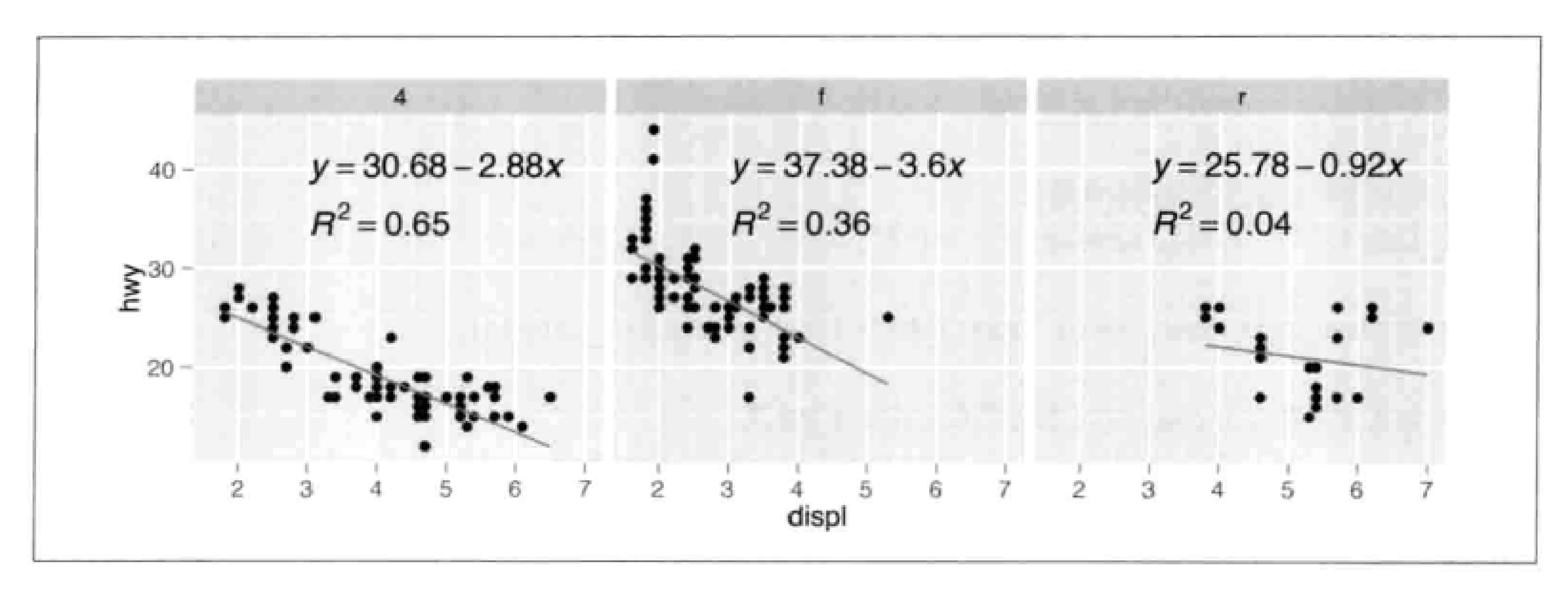


图 7-18　各个分面中的注解，含有关于数据的信息

我们需要在这里编写自己的函数，是因为要生成线性模型并提取系数需要直接操作数据框中的每个子集。如果你仅仅想要展示 r^2 值，通过配合 ddply() 使用 summarise() 函数并传递附加参数给 summarise()，完全可以做得更简单：

```
# 计算每组的 r^2 值
labels <- ddply(mpg, "drv", summarise, r2 = cor(displ, hwy)^2)
labels$r2 <- sprintf("italic(R^2) == %.2f", labels$r2)
```

文本类几何对象并不是可向每个分面独立添加的唯一几何对象。只要输入的数据结构正确，我们可以使用任意几何对象。

另见

参见 7.2 节以了解更多在图形中使用数学表达式的方法。

如果你不想让 ggplot2 使用 stat_smooth() 为你绘制预测曲线，而是希望使用自己的模型对象绘制，请参见 5.8 节。

第8章

坐标轴

x 轴和 *y* 轴为解读所展示的数据提供了上下文信息。ggplot2 以默认设置显示的坐标轴在多数情况下看起来都不错，不过你也许希望能够控制细节，例如，坐标轴标签、刻度线的数量和布局、刻度线标签等。在本章中，将介绍如何微调坐标轴的外观。

8.1 交换 *x* 轴和 *y* 轴

问题

如何交换一幅图上的 *x* 轴和 *y* 轴？

方法

使用 coord_flip() 来翻转坐标轴（见图 8-1）：

```
ggplot(PlantGrowth, aes(x=group, y=weight)) + geom_boxplot()

ggplot(PlantGrowth, aes(x=group, y=weight)) + geom_boxplot() + coord_flip()
```

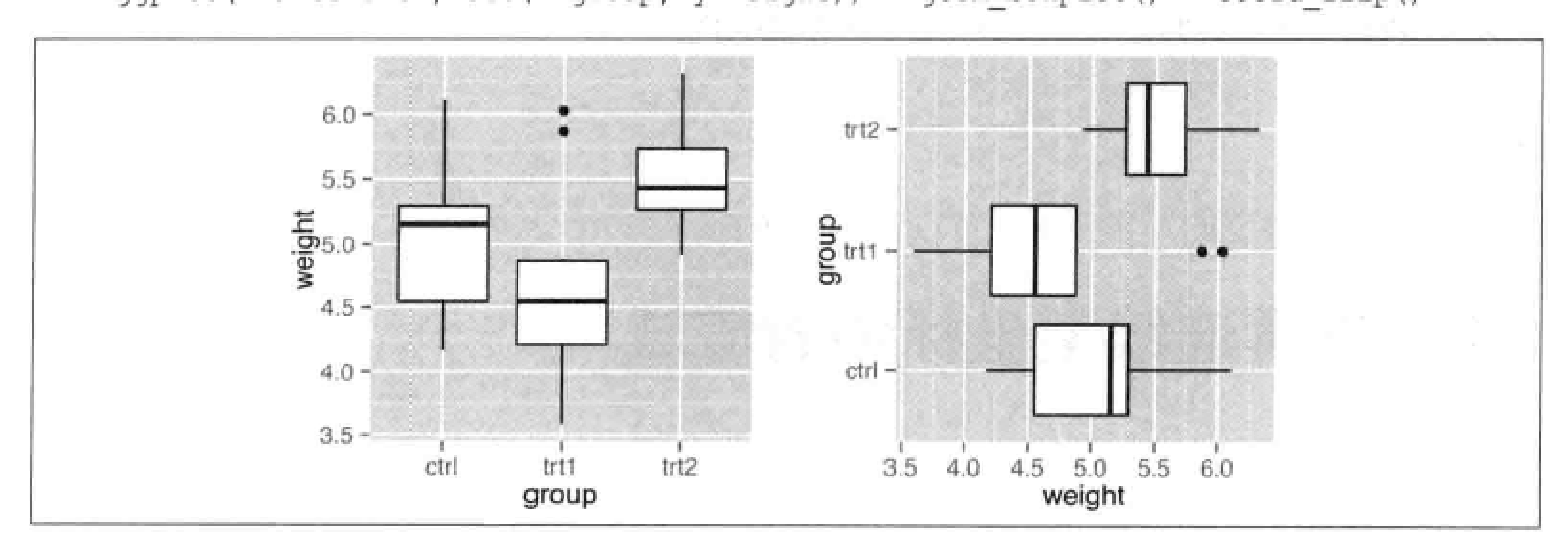


图 8-1 左图：含常规坐标轴的箱线图 右图：交换了坐标轴的箱线图

讨论

对于散点图来说，调换纵轴和横轴上显示的元素非常简单：仅仅交换映射到 x 和 y 的变量就可以了。但并不是所有 ggplot2 中的几何对象都会同等对待 *x* 轴和 *y* 轴。举例来说，箱线图依 *y* 轴对数据计算统计摘要，折线图中的线段只沿 *x* 轴移动，误差线只有一个单独的 *x* 值但具有若干 *y* 值，等等。如果你正在使用这些几何对象，并且希望在图形中交换它们的坐标轴，那么 `coord_flip()` 正是你所需要的。

有时在交换坐标轴后，各项的顺序可能正好与你想要的相反。在一幅有着标准 *x* 轴和 *y* 轴的图形上，与 *x* 对应的项目从左到右排列，这与正常从左到右的阅读方式一致。但是当你交换了坐标轴，各项仍是从原点开始向外排列，在这种情况下就是从下到上，与正常从上到下的阅读方式发生冲突。某些时候这是一个问题，某些时候又不是。如果 x 变量是一个因子型变量，则排列顺序可以通过使用 `scale_x_discrete()` 和参数 `limits=rev(levels(...))` 进行反转，如图 8-2 所示：

```
ggplot(PlantGrowth, aes(x=group, y=weight)) + geom_boxplot() + coord_flip() +
    scale_x_discrete(limits=rev(levels(PlantGrowth$group)))
```

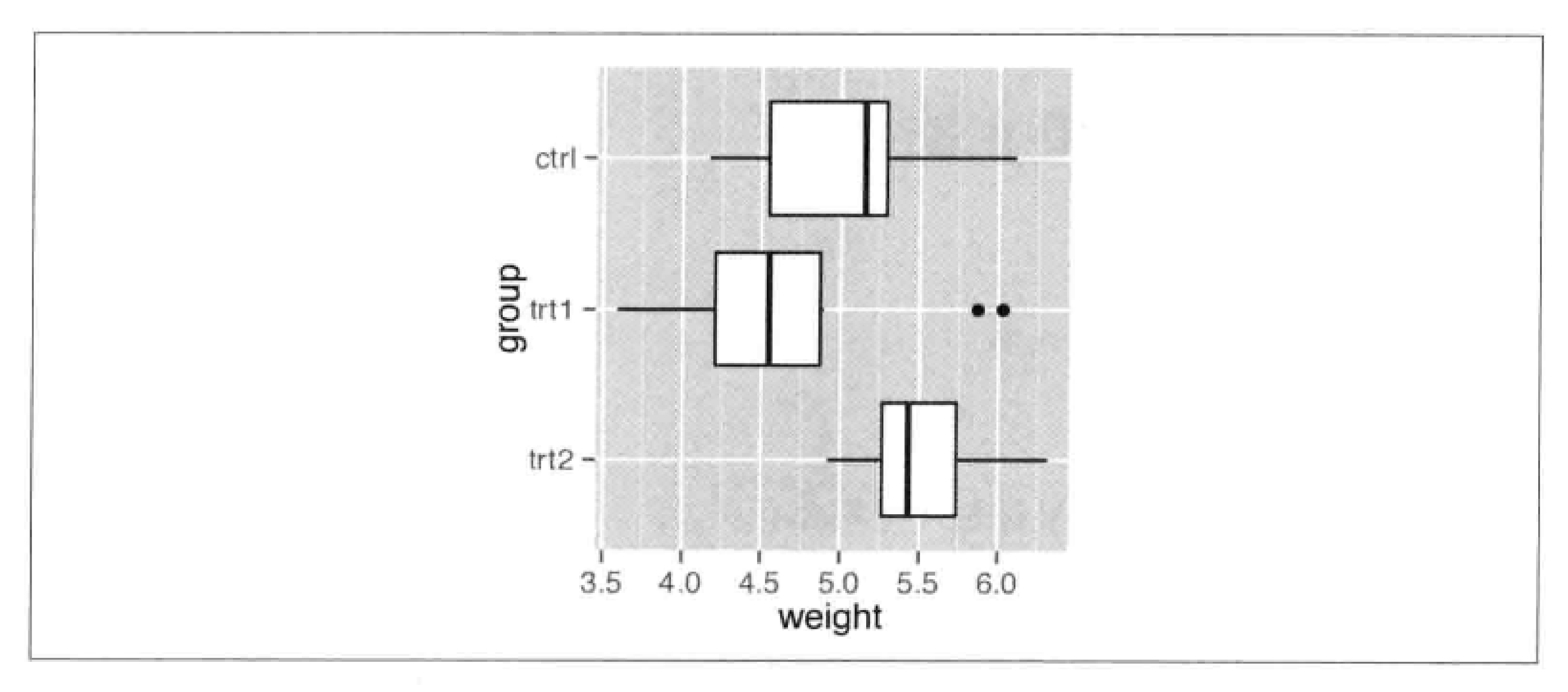


图 8-2　交换了坐标轴并反转了 *x* 轴元素顺序的箱线图

另见

如果变量是连续型的，参见 8.3 节以反转其方向。

8.2　设置连续型坐标轴的值域

问题

如何设置某条坐标轴的值域（或范围）？

方法

你可以使用 xlim() 或 ylim() 来设置一条连续型坐标轴的最小值和最大值。图 8-3 展示了一幅使用默认 y 轴范围的图形和另一幅手动设定 y 轴范围的图形：

```
p <- ggplot(PlantGrowth, aes(x=group, y=weight)) + geom_boxplot()
# 显示基本图形
p

p + ylim(0, max(PlantGrowth$weight))
```

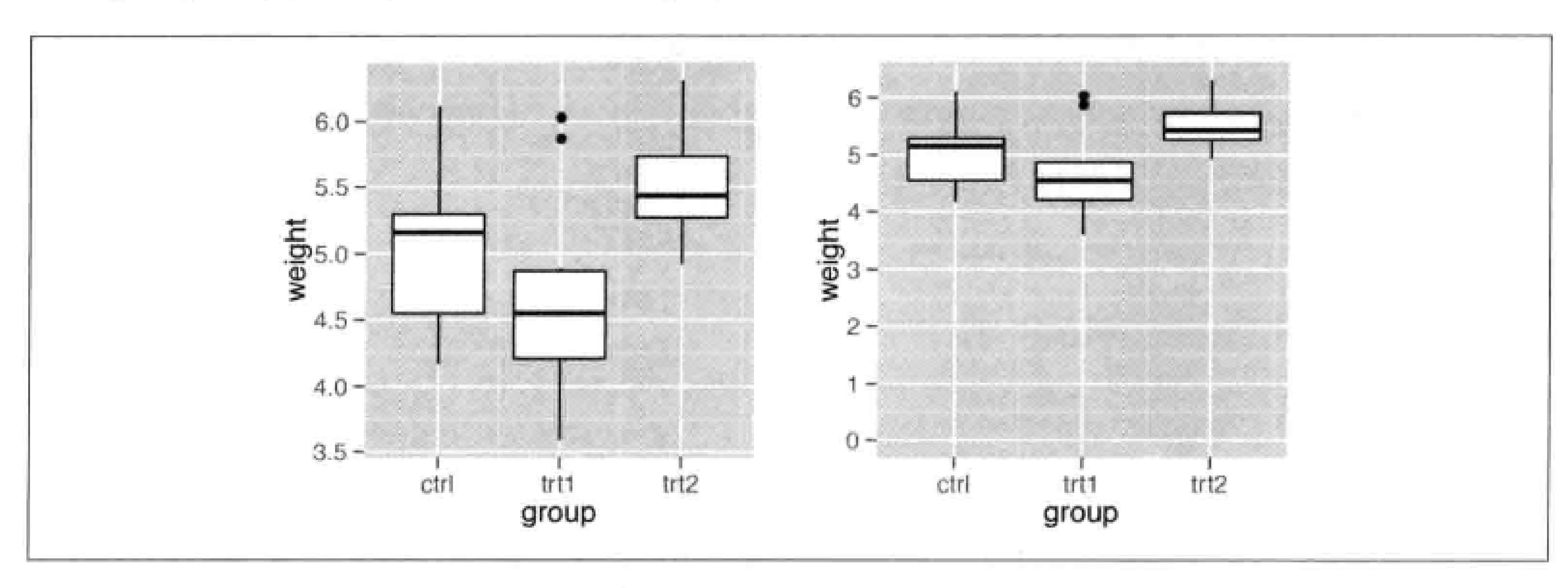


图 8-3　左图：使用默认值域的箱线图　右图：使用手动设定值域的箱线图

第二个示例将 y 轴的值域设置为从 0 到 weight 列的最大值，当然此处使用一个常数值（如 10）作为最大值也无妨。

讨论

使用 ylim() 来设定范围是通过 scale_y_continuous() 来设定范围的简便写法（对于 xlim() 和 scale_x_continuous() 同理）。以下两种表达方式等价：

```
ylim(0, 10)
scale_y_continuous(limits=c(0, 10))
```

有时，你需要设定 scale_y_continuous() 的其他属性，在这些情况下同时使用 ylim() 和 scale_y_continuous() 可能会让程序产生一些不可预知的行为，这是因为只有命令中的后一条会生效。在以下两个示例中，ylim(0, 10) 应当设定 y 的值域为从 0 到 10，而 scale_y_continuous(breaks=c(0, 5, 10)) 应将刻度线放置到 0、5、10 的位置。但是在这两个例子中，仅有第二条命令生效[1]：

```
p + ylim(0, 10) + scale_y_continuous(breaks=c(0, 5, 10))

p + scale_y_continuous(breaks=c(0, 5, 10)) + ylim(0, 10)
```

① 原书中给出的代码为 breaks=NULL，与文中所述 breaks=c(0,5,10) 不一致，可能是笔误。本段和下一段中含有 breaks 的三条命令已经依照原文进行了修正。——译者注

要让两项修改均生效，舍弃 ylim() 并直接在 scale_y_continuous() 中同时设定 limits 和 breaks 即可：

```
p + scale_y_continuous(limits=c(0, 10), breaks=c(0, 5, 10))
```

ggplot2 中有两种设置坐标轴值域的方式。第一种方式是修改标度，第二种方式是应用一个坐标变换。当你修改 x 标度和 y 标度的范围时，任何在范围以外的数据都会被移除——换言之，超出范围的数据不仅不会被展示，而且会被完全移出考虑处理的数据范围。

以上文的箱线图为例，如果你限制了 y 的值域，使得某些原始数据被剪除掉，则箱线图中统计量的计算都会基于修剪后的数据，而箱线的形状也会随之改变。

通过使用坐标变换，数据则不会被修剪；从本质上说，它只是将数据放大或缩小到指定的范围。图 8-4 展示了两种方式的区别：

```
p + scale_y_continuous(limits = c(5, 6.5)) # 与使用 ylim() 相同

p + coord_cartesian(ylim = c(5, 6.5))
```

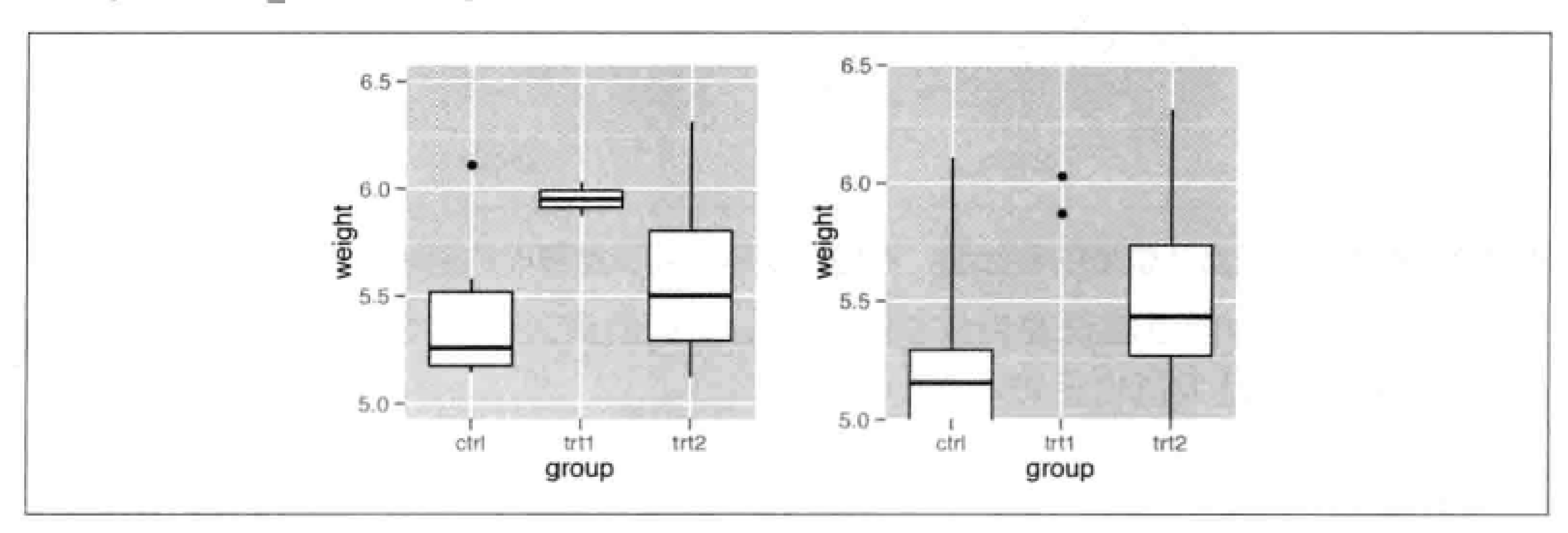


图 8-4　左图：使用标度限制 y 到更小的范围（数据被丢弃，所以箱线图的形状发生了改变）右图：使用坐标变换“放大”了数据

最后，通过使用 expand_limits() 来单向扩展值域也是可以的（见图 8-5）。不过，你不能使用它来缩减值域：

```
p + expand_limits(y=0)
```

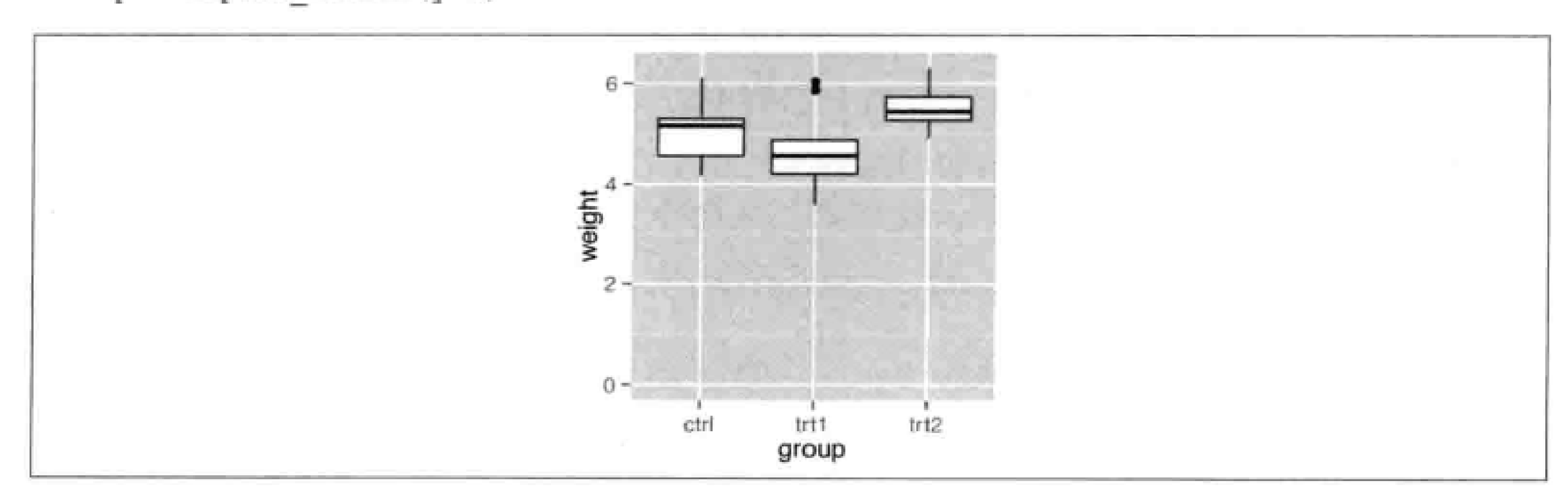


图 8-5　y 的值域被扩展到包含 0 的箱线图

8.3 反转一条连续型坐标轴

问题

如何反转一条连续型坐标轴的方向？

方法

使用 scale_y_reverse 或 scale_x_reverse（见图 8-6）。坐标轴的方向也可通过指定反序的范围来反转，先写最大值，再写最小值：

```
ggplot(PlantGrowth, aes(x=group, y=weight)) + geom_boxplot() + scale_y_reverse()

# 通过指定反序的范围产生类似的效果
ggplot(PlantGrowth, aes(x=group, y=weight)) + geom_boxplot() + ylim(6.5, 3.5)
```

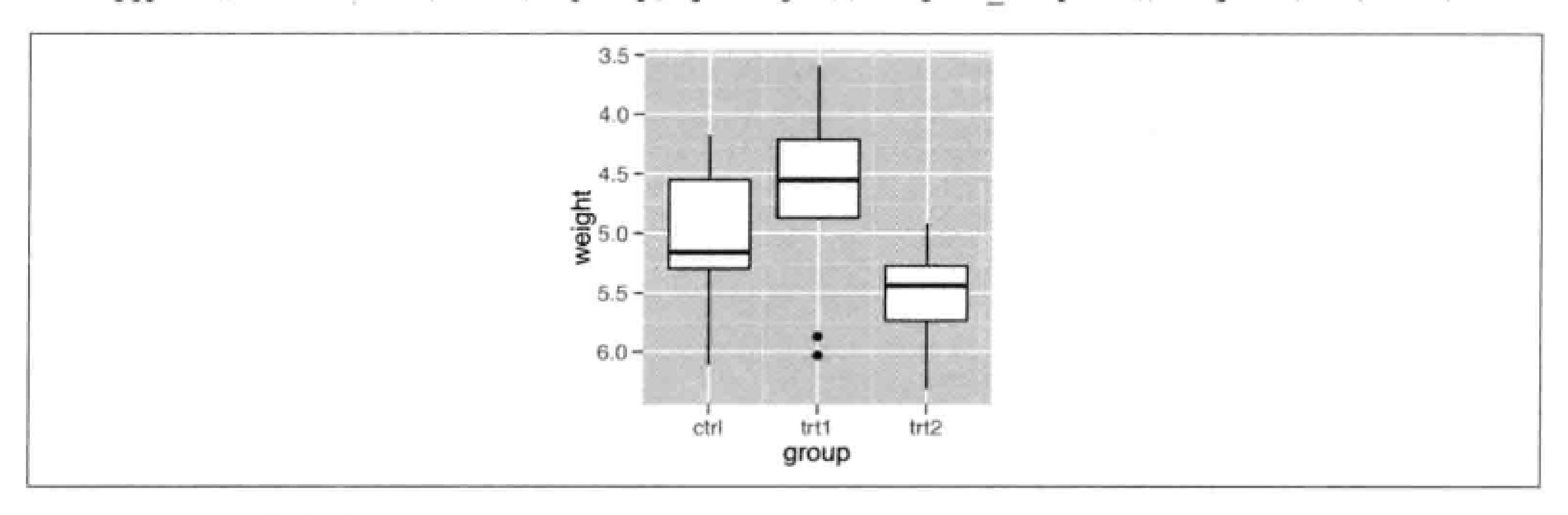


图 8-6　反转 y 轴后的箱线图

讨论

与 scale_y_continuous() 类似，scale_y_reverse() 也无法与 ylim 配合工作（对 x 轴属性也一样）。如果你希望反转某条坐标轴并为它设定值域，则必须通过反序设定范围的方式，在 scale_y_reverse() 语句内完成（见图 8-7）：

```
ggplot(PlantGrowth, aes(x=group, y=weight)) + geom_boxplot() +
    scale_y_reverse(limits=c(8, 0))
```

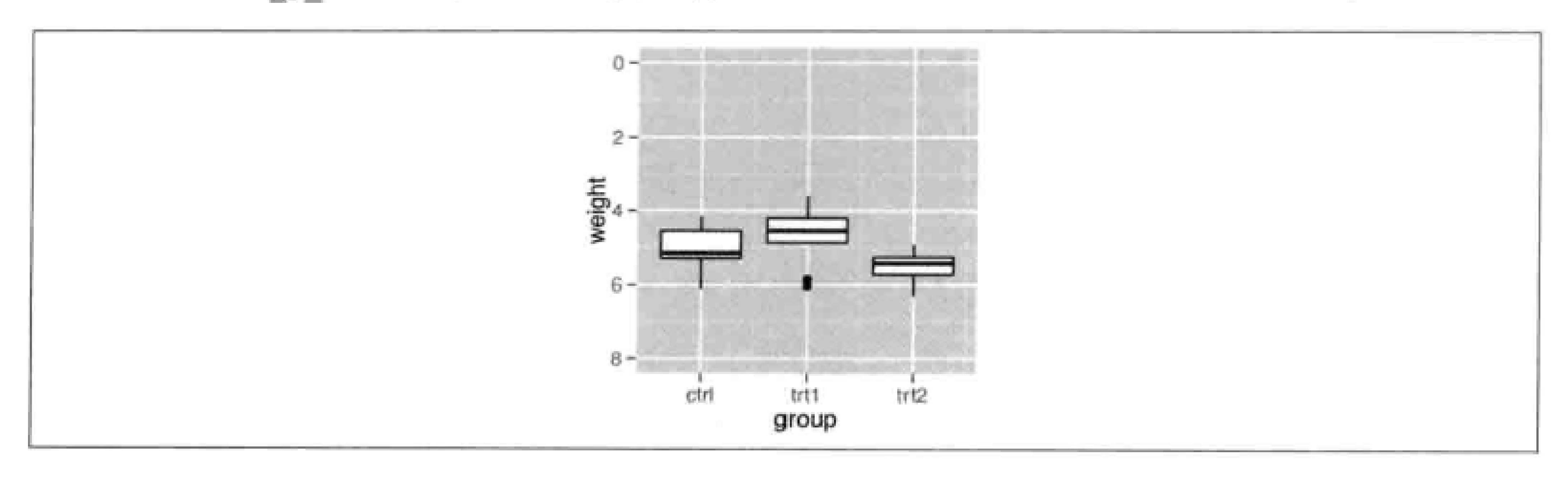


图 8-7　手动设定范围且反转了 y 轴的箱线图

另见

要反转离散型坐标轴项目的顺序，参见 8.4 节。

8.4 修改类别型坐标轴上项目的顺序

问题

如何修改类别型坐标轴上项目的顺序？

方法

对于类别型（或者说离散型）坐标轴来说，会有一个因子型变量映射到它上面，坐标轴上项目的顺序可以通过设定 scale_x_discrete() 或 scale_y_discrete() 中的参数 limits 来修改。

要手动设定坐标轴上项目的顺序，将一个依理想顺序排列的水平向量指定给 limits 即可。你也可以使用这个向量来忽略某些项目，如图 8-8 所示：

```
p <- ggplot(PlantGrowth, aes(x=group, y=weight)) + geom_boxplot()

p + scale_x_discrete(limits=c("trt1","ctrl","trt2"))
```

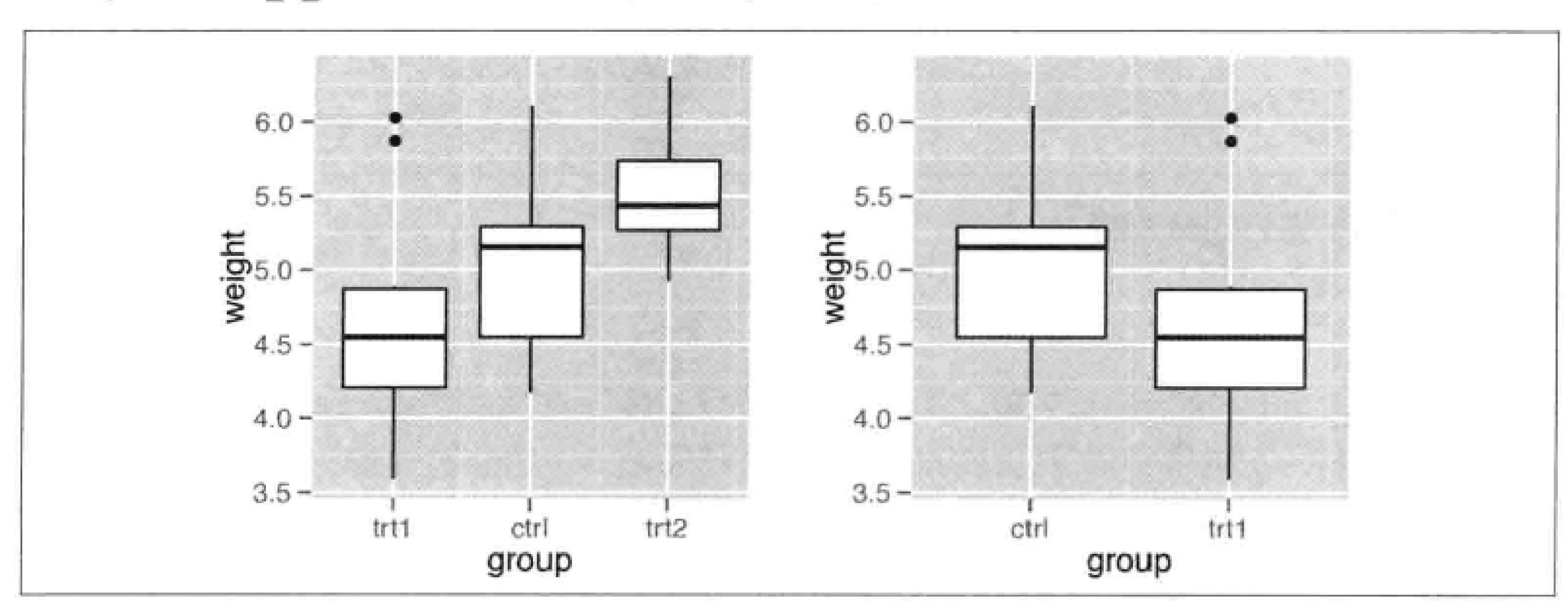


图 8-8　左图：手动指定 x 轴项目顺序的箱线图　右图：仅保留两项的箱线图

讨论

你也可以使用以上方法在坐标轴上展示项目的子集。使用以下语句将仅显示 ctrl 和 trt1（见图 8-8 右图）：

```
p + scale_x_discrete(limits=c("ctrl","trt1"))
```

要反转项目顺序，设定 limits=rev(levels(...))，将因子型变量放入括号中即可。以下语句将反转因子 PlantGrowth$group 的顺序，如图 8-9 所示：

```
p + scale_x_discrete(limits=rev(levels(PlantGrowth$group)))
```

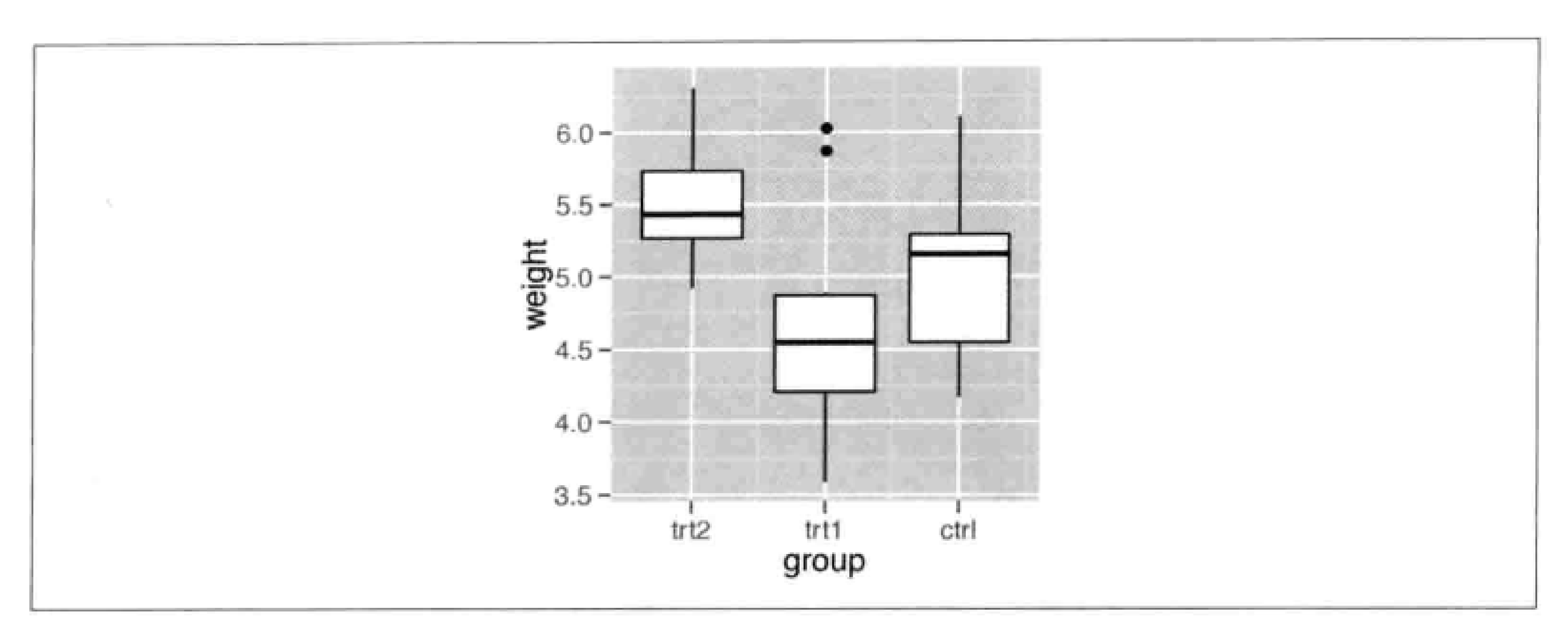


图 8-9　反转了 x 轴项目顺序的箱线图

另见

要根据另外一列数据的值对因子水平进行重排序，参见 15.9 节。

8.5　设置 x 轴和 y 轴的缩放比例

问题

如何设置 x 轴和 y 轴的缩放比例？

方法

使用 `coord_fixed()`。以下代码将得到 x 轴和 y 轴之间 1:1 的缩放结果[①]，如图 8-10 所示：

```
library(gcookbook) # 为了使用数据集

sp <- ggplot(marathon, aes(x=Half,y=Full)) + geom_point()

sp + coord_fixed()
```

讨论

`marathon` 数据集中包含了跑步者的全程马拉松成绩和半程马拉松成绩。在这种情况下，强制相同的 x 轴和 y 轴缩放比例可能是有用的。

通过在 `scale_y_continuous()` 和 `scale_x_continuous()` 中调整参数 `breaks`，从而将刻度间距设为相同，也会有所帮助（同样见图 8-10）：

① 作者在这里所指的 1:1 缩放并不是缩放结果的总长宽比例为 1:1，而是指坐标轴单位长度表示的数值范围是 1:1 的。默认情况下，ggplot2 使两轴总长宽比例为 1:1，从而形成正方形的绘图区域，此时两轴上单位长度表示的数值范围往往是不同的。——译者注

```
sp + coord_fixed() +
    scale_y_continuous(breaks=seq(0, 420, 30)) +
    scale_x_continuous(breaks=seq(0, 420, 30))
```

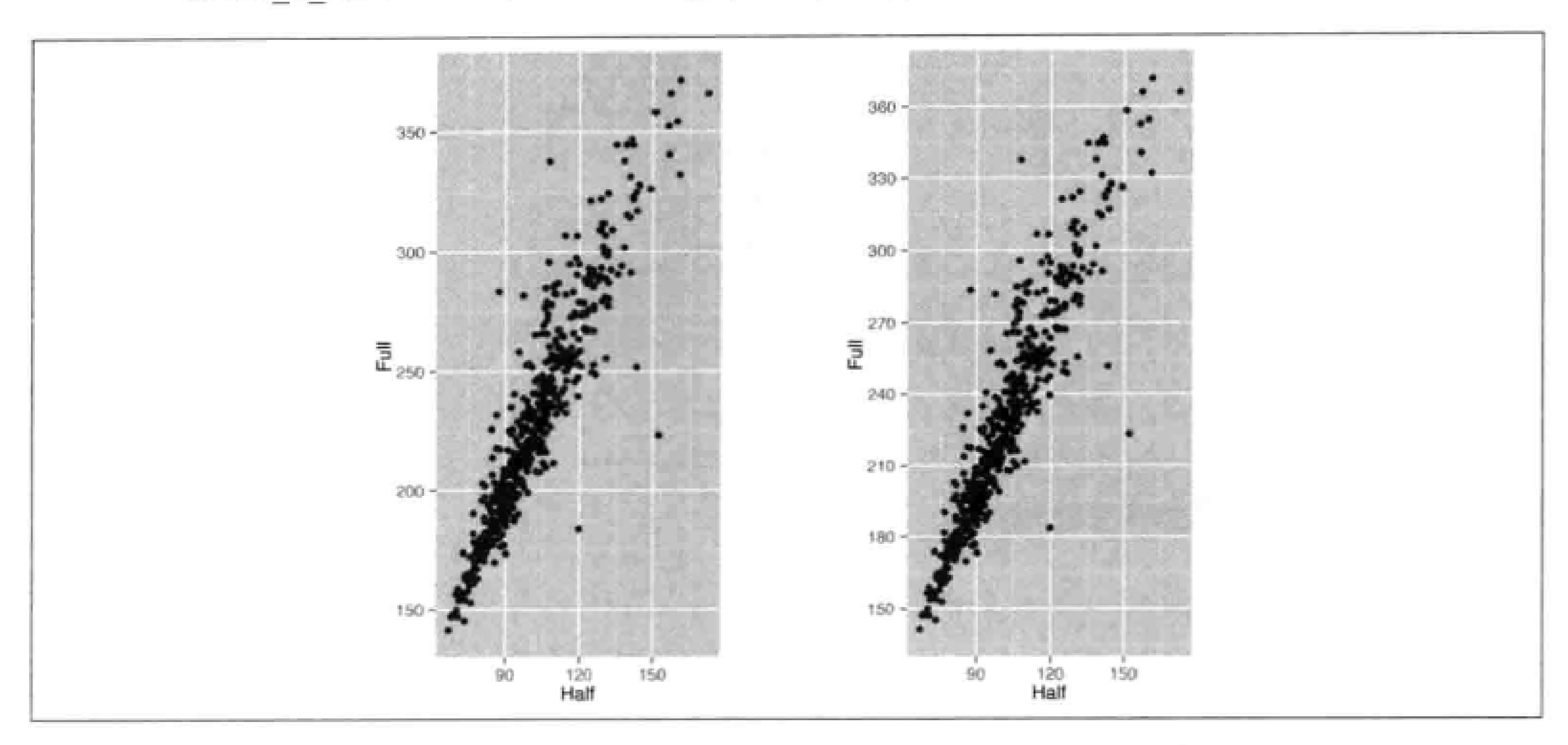


图 8-10　左图：等长缩放坐标轴的散点图　右图：在指定位置放置了刻度线的散点图

如果你希望为两个坐标轴之间指定其他的固定比例而非相同的比例，可以设置参数 `ratio`。对于 `marathon` 数据集，我们可能想让对应半程马拉松时间的坐标轴被拉伸到全程马拉松时间坐标轴的两倍①（见图 8-11）。我们也将在 *x* 轴上添加双倍的刻度线：

```
sp + coord_fixed(ratio=1/2) +
    scale_y_continuous(breaks=seq(0, 420, 30)) +
    scale_x_continuous(breaks=seq(0, 420, 15))
```

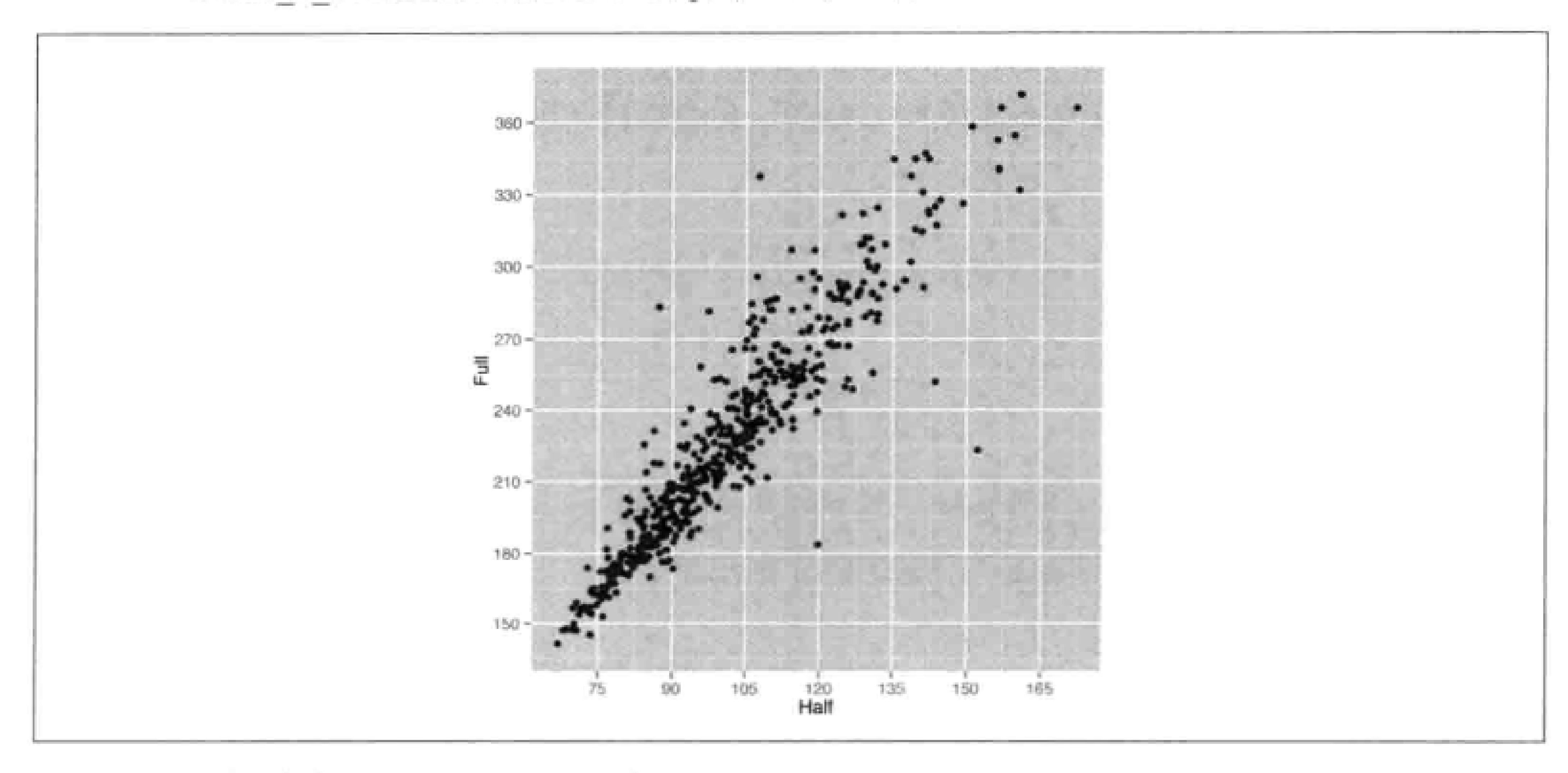


图 8-11　坐标轴缩放比例为 1/2 的散点图

① 这里的倍数同样指单位长度坐标轴表示的数值范围，同上。——译者注

8.6 设置刻度线的位置

问题

如何设置刻度线在坐标轴上出现的位置？

方法

通常来说 ggplot() 会自动将刻度线摆放在合适的位置，但如果你希望改变它们的位置，设置标度中的参数 breaks 即可（见图 8-12）：

```
ggplot(PlantGrowth, aes(x=group, y=weight)) + geom_boxplot()

ggplot(PlantGrowth, aes(x=group, y=weight)) + geom_boxplot() +
    scale_y_continuous(breaks=c(4, 4.25, 4.5, 5, 6, 8))
```

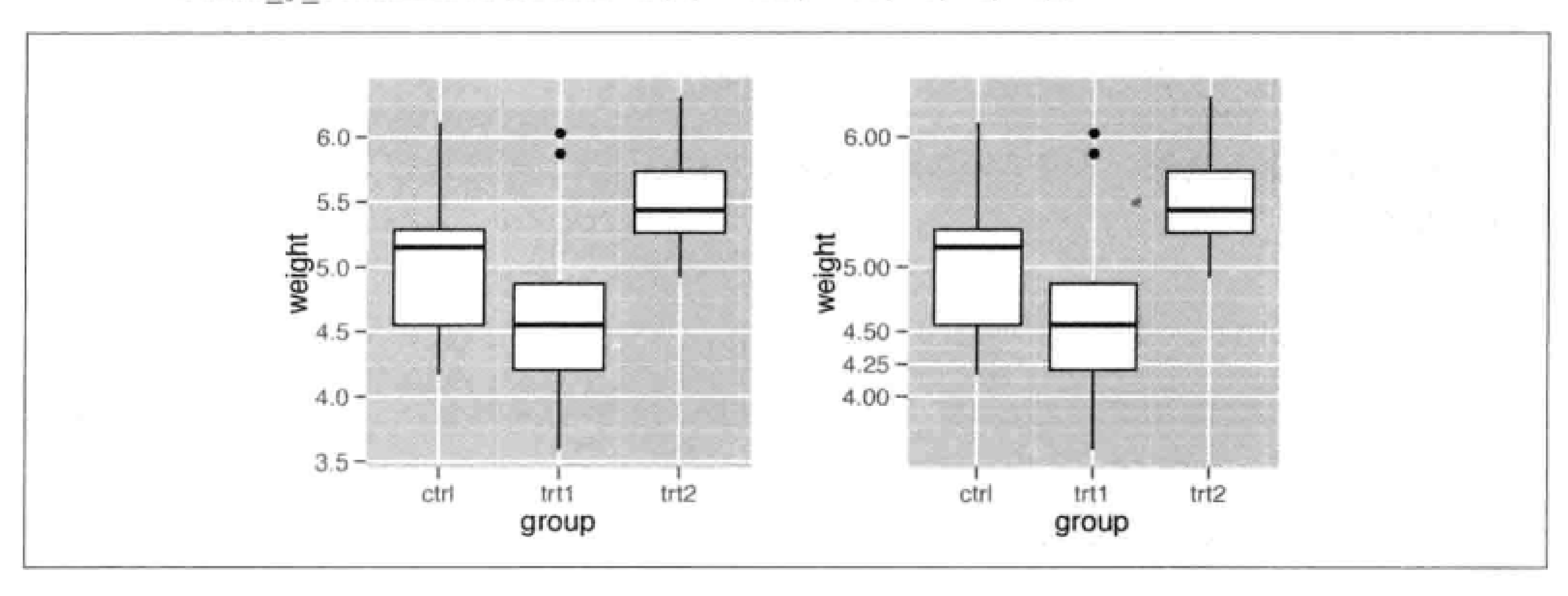


图 8-12　左图：自动确定刻度线的箱线图　右图：手动设置刻度线的箱线图

讨论

刻度线的位置决定了绘制主网格线的位置。如果该坐标轴表示一个连续型变量，那么颜色更暗且没有标签的次网格线将被默认绘制在每两个主网格线的正中间位置。

你也可以使用 seq() 函数或运算符 : 来生成刻度线的位置向量：

```
seq(4, 7, by=.5)

4.0 4.5 5.0 5.5 6.0 6.5 7.0

5:10

5  6  7  8  9 10
```

如果坐标轴是离散型而不是连续型的，则默认会为每个项目生成一条刻度线。对于离散型坐标轴，你可以通过指定 limits 来修改项目的顺序或移除项目（参见 8.4 节）。设定 breaks 将会决定为哪些水平加上标签，但不会移除它们或是改变它们的顺序。图

8-13 展示了当你设定 limits 和 breaks 时将会发生的情况：

```
# 为离散型坐标轴同时设定 breaks 和 limits
ggplot(PlantGrowth, aes(x=group, y=weight)) + geom_boxplot() +
    scale_x_discrete(limits=c("trt2", "ctrl"), breaks="ctrl")
```

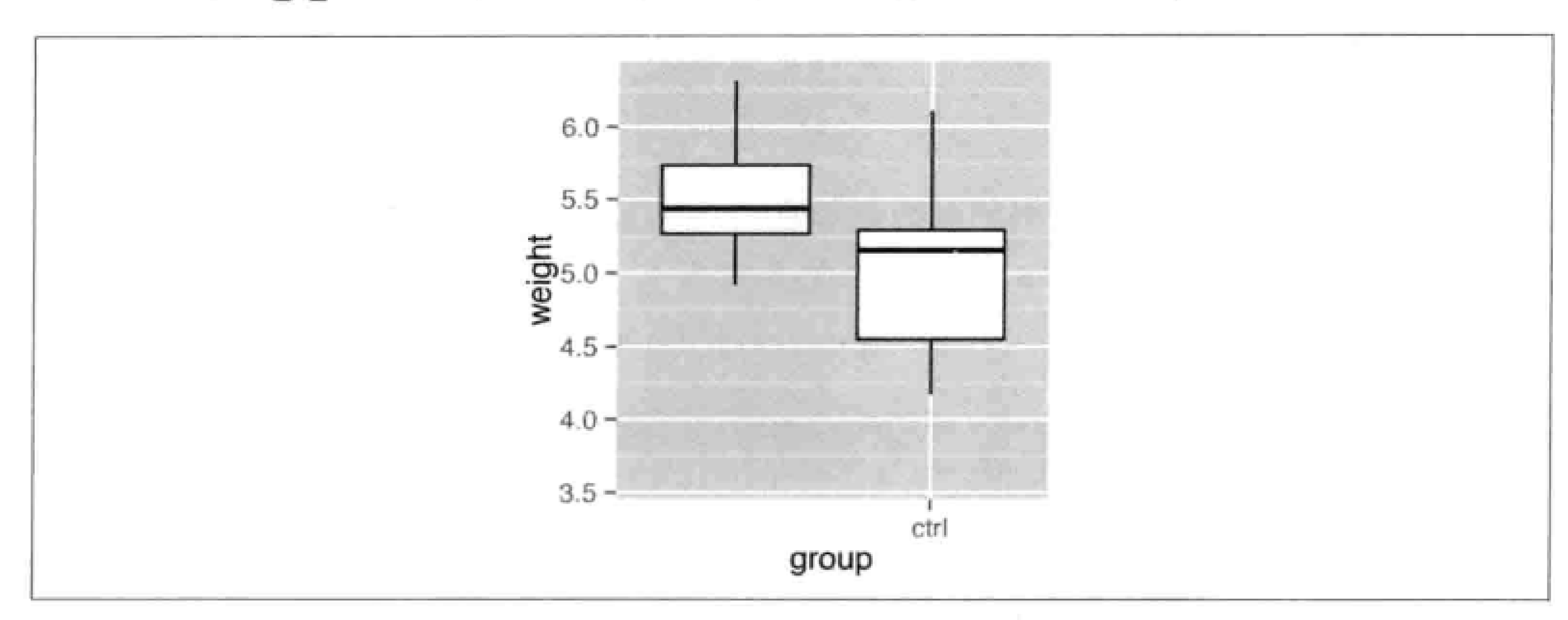


图 8-13　对离散型坐标轴，设置 limits 以重排序或移除项目，同时设置 breaks 来控制哪些项目拥有标签

另见

要从图中移除刻度线和刻度线标签（不修改数据），参见 8.7 节。

8.7　移除刻度线和标签

问题

如何移除刻度线和刻度标签？

方法

要像图 8-14 左图一样仅移除刻度标签，使用 theme(axis.text.y = element_blank())（也可对 axis.text.x 做相同处理）即可。这种方法对于连续型和离散型坐标轴均有效：

```
p <- ggplot(PlantGrowth, aes(x=group, y=weight)) + geom_boxplot()

p + theme(axis.text.y = element_blank())
```

要移除刻度线，可使用 theme(axis.ticks=element_blank())。这样将会同时移除两轴的刻度线（无法仅隐藏单个坐标轴的刻度线）。在本例中，我们将隐藏所有的刻度线和 y 轴的刻度标签（见图 8-14 中图）：

```
p + theme(axis.ticks = element_blank(), axis.text.y = element_blank())
```

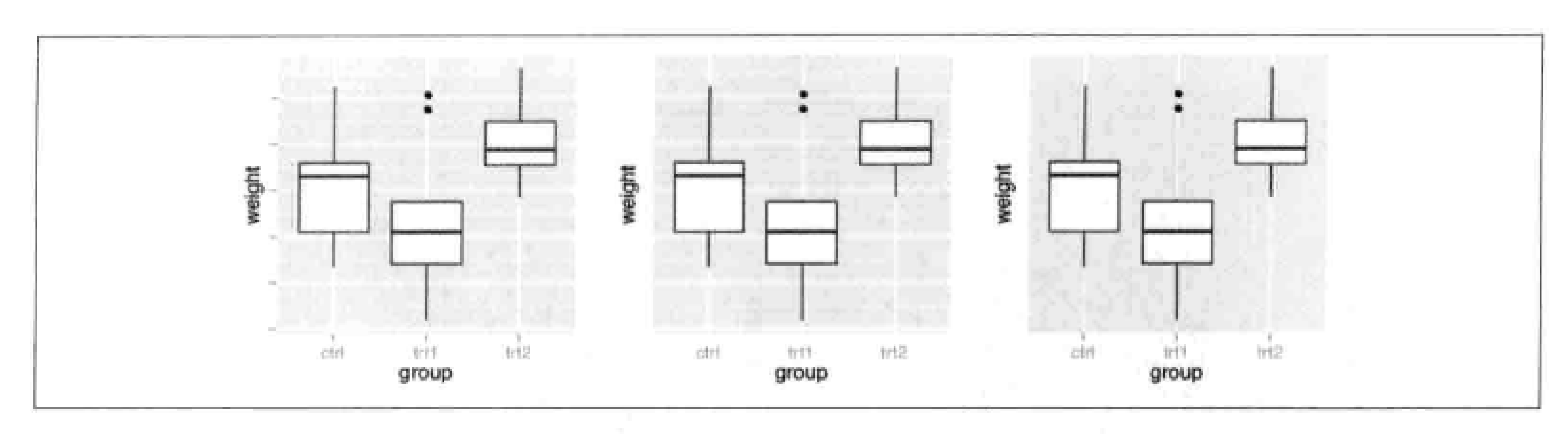


图 8-14　左图：*y* 轴上无刻度标签；中图：*y* 轴上既无刻度线也无刻度标签　右图：使用 **breaks=NULL**

要移除刻度线、刻度标签和网格线，将 breaks 设置为 NULL 即可（见图 8-14 右图）：

```
p + scale_y_continuous(breaks=NULL)
```

这种方法仅对连续型坐标轴有效；如果像 8.4 节中那样使用 limits 从类别型坐标轴上移除项目，则含有对应值的数据将完全不被显示。

讨论

事实上，共有三种项目可以控制：刻度标签、刻度线和网格线。对于连续型坐标轴，ggplot() 通常会在每个 breaks 值的位置放置刻度线、刻度标签和主网格线。对于类别型坐标轴，这些元素则出现在每个 limits 值的位置。

我们可以独立控制每条坐标轴上的刻度标签。但是，刻度线和网格线必须同时控制。

8.8　修改刻度标签的文本

问题

如何修改刻度标签的文本？

方法

考虑图 8-15 中的散点图，身高（变量 heightIn）是以英寸的数值表示的：

```
library(gcookbook) # 为了使用数据集

hwp <- ggplot(heightweight, aes(x=ageYear, y=heightIn)) +
         geom_point()

hwp
```

要像图 8-15 右图一样任意设定标签，在标度中为 breaks 和 labels 赋值即可。其中的一个标签含有一个换行符（\n），意为让 ggplot() 在那里另起一行：

```
hwp + scale_y_continuous(breaks=c(50, 56, 60, 66, 72),
                         labels=c("Tiny", "Really\nshort", "Short",
                                  "Medium", "Tallish"))
```

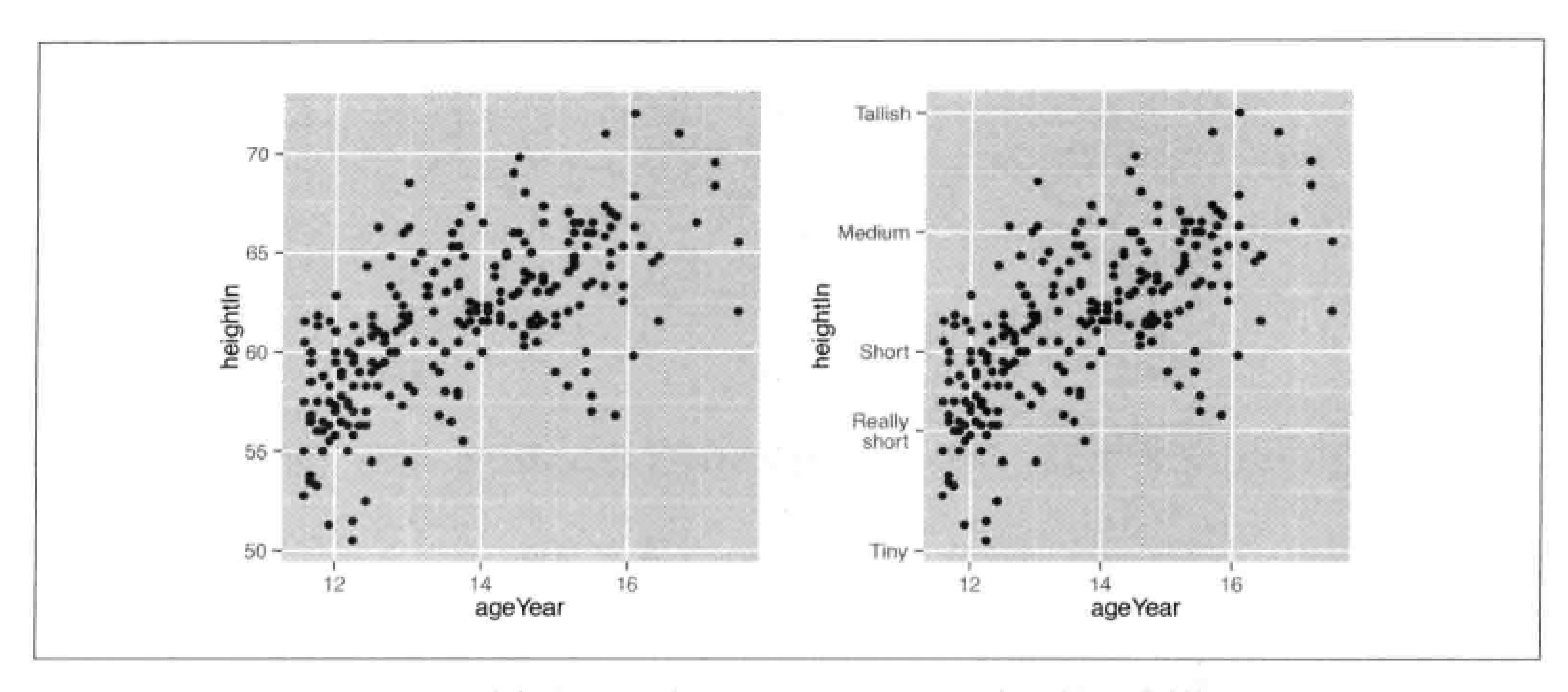


图 8-15　左图：自动绘制刻度标签的散点图　右图：手动指定 y 轴刻度标签的散点图

讨论

除了完全任意地设置标签以外，更常见的情况是数据以某种格式存储，而我们希望以另外一种格式显示标签。举例来说，我们可能想让身高变量显示为英尺和英寸的格式（像 5′6″这样），而不是仅仅显示一个英寸数值。要完成这项任务，我们可以定义一个格式刷（formatter）函数，这样的函数可以读入数值并返回相应的字符串。例如，以下函数可将英寸数值转换为英尺加英寸的格式：

```
footinch_formatter <- function(x) {
    foot <- floor(x/12)
    inch <- x %% 12
    return(paste(foot, "'", inch,"\"", sep=""))
}
```

下面是此函数对输入值 56 ～ 64 的返回结果（反斜杠是转义符，用来区分字符串中所含的引号和字符串本身的定界引号）：

```
footinch_formatter(56:64)
```

```
"4'8\"" "4'9\"" "4'10\"" "4'11\"" "5'0\"" "5'1\"" "5'2\"" "5'3\"" "5'4\""
```

现在就可以使用参数 labels 把我们的函数传递给标度了（见图 8-16）：

```
hwp + scale_y_continuous(labels=footinch_formatter)
```

在图中，每隔五英寸放置了一个自动生成的刻度线，但是对于这个数据来说看起来有些古怪。我们可以通过指定参数 breaks 让 ggplot() 每隔四英寸设置一条刻度线取而代之（见图 8-16 右图）：

```
hwp + scale_y_continuous(breaks=seq(48, 72, 4), labels=footinch_formatter)
```

另一项常见任务是将时间测度转换为 HH:MM:SS（时 : 分 : 秒）或者其他类似的格式。以下函数可以读入分钟的数值并将它们转换为这种格式，同时舍入到最接近的秒数

（也可以按照你的特殊需要来自定义）：

```
timeHMS_formatter <- function(x) {
    h <- floor(x/60)
    m <- floor(x %% 60)
    s <- round(60*(x %% 1))                    # 舍入到最接近的秒数
    lab <- sprintf("%02d:%02d:%02d", h, m, s)  # 格式化字符串为 HH:MM:SS 的格式
    lab <- gsub("^00:", "", lab)               # 如果开头存在 00: 则移除
    lab <- gsub("^0", "", lab)                 # 如果开头存在 0  则移除
    return(lab)
}
```

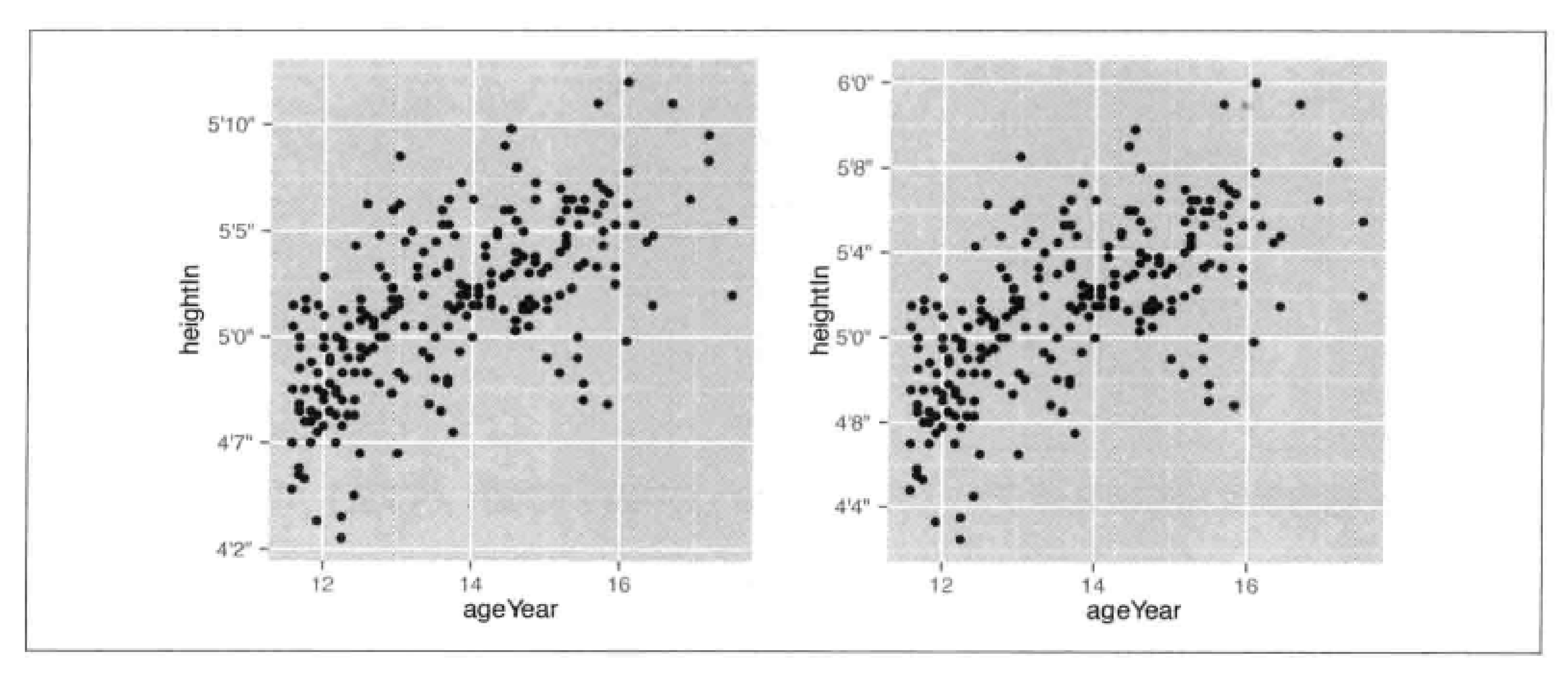


图 8-16　左图：使用格式刷函数的散点图　右图：手动指定 *y* 轴 breaks 的散点图

使用一些示例数值运行它会得到：

```
timeHMS_formatter(c(.33, 50, 51.25, 59.32, 60, 60.1, 130.23))

"0:20"   "50:00"   "51:15"   "59:19"   "1:00:00" "1:00:06" "2:10:14"
```

随 ggplot2 安装的 scales 包自带了一些内建的格式化函数：

- comma() 在千、百万、十亿等位置向数字添加逗号[①]。
- dollar() 添加一个美元符号并舍入到最接近的美分。
- percent() 乘以 100，舍入到最接近的整数值，并添加一个百分号。
- scientific() 对大数字和小数字给出科学记数法表示，如 3.30e+05。

如果你希望使用这些函数，必须首先使用 library(scales) 加载 scales 包。

8.9　修改刻度标签的外观

问题

如何修改刻度标签的外观？

① 即英语中对于阿拉伯数字三位一隔的记法。——译者注

方法

在图 8-17 左图中，我们手动设定了较长的标签——长到足以互相重叠：

```
bp <- ggplot(PlantGrowth, aes(x=group, y=weight)) + geom_boxplot() +
      scale_x_discrete(breaks=c("ctrl", "trt1", "trt2"),
                       labels=c("Control", "Treatment 1", "Treatment 2"))
bp
```

要将文本逆时针旋转 90°（见图 8-17 中图），只需使用：

```
bp + theme(axis.text.x = element_text(angle=90, hjust=1, vjust=.5))
```

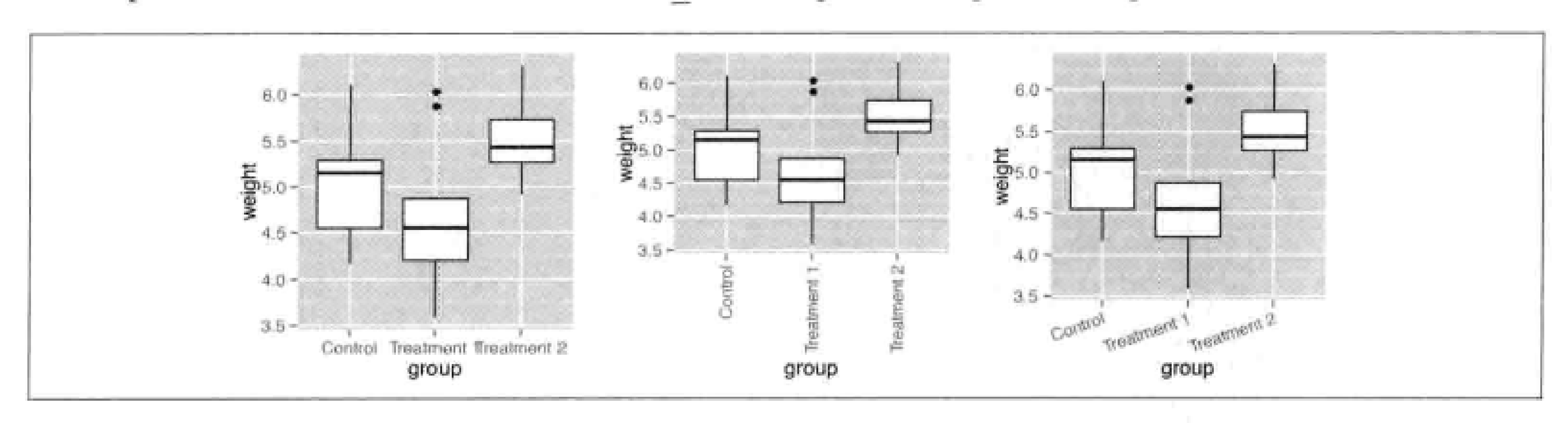


图 8-17　*x* 轴的刻度标签被旋转了 0°（左图）、90°（中图）和 30°（右图）

将文本旋转 30°（见图 8-17 右图）可以占用更小的纵向空间，并且可以让你在不转头的情况下还能容易地阅读：

```
bp + theme(axis.text.x = element_text(angle=30, hjust=1, vjust=1))
```

参数 hjust 和 vjust 设置了横向对齐（左对齐 / 居中 / 右对齐）和纵向对齐（顶部对齐 / 居中 / 底部对齐）。

讨论

除了旋转以外，其他的文本属性，如大小、样式（粗体 / 斜体 / 常规）和字体族（如 Times 或 Helvetica）可以使用 element_text() 进行设置，如图 8-18 所示：

```
bp + theme(axis.text.x = element_text(family="Times", face="italic",
                                       colour="darkred", size=rel(0.9)))
```

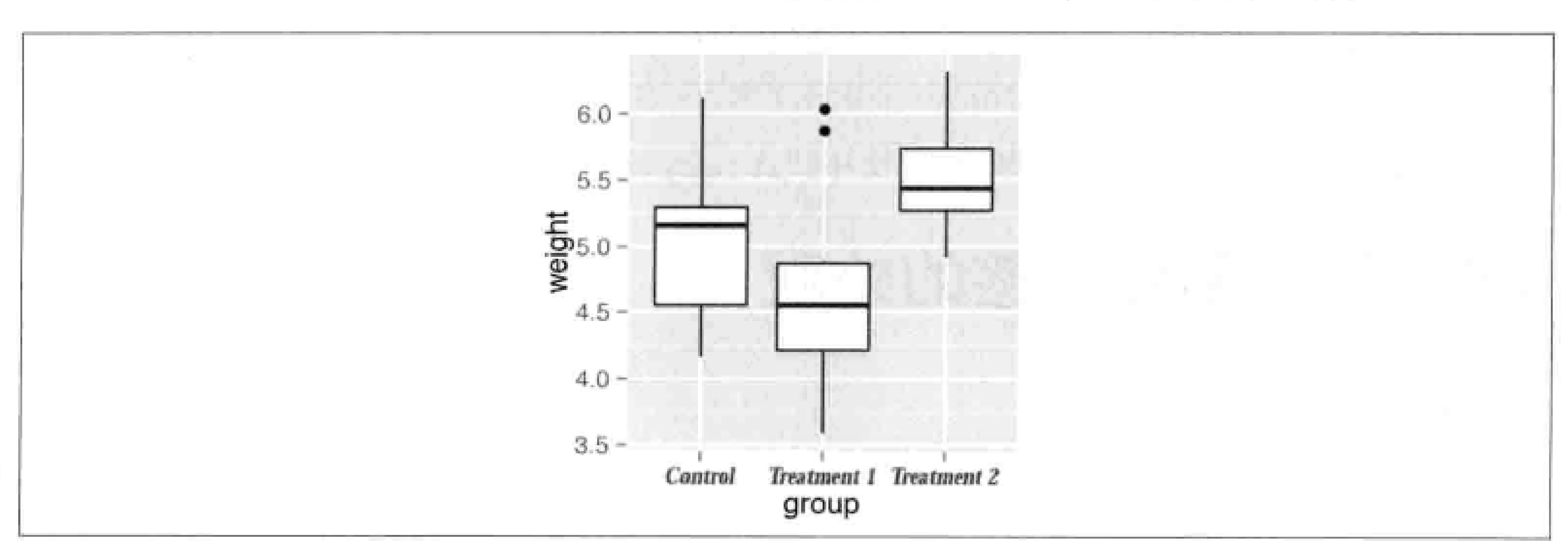


图 8-18　手动指定外观的 *x* 轴刻度标签

在本例中，size（文字大小）被设为 rel(0.9)，意为当前主题基础字体大小的 0.9 倍。

这些命令仅仅控制了单个坐标轴上刻度标签的外观，并不影响其他坐标轴、坐标轴标签、整体的标题或图例。要同时控制所有这些元素的外观，可以使用主题系统，如 9.3 节中所讨论的那样。

另见

参见 9.2 节以了解更多关于如何控制文本外观的信息。

8.10 修改坐标轴标签的文本

问题

如何修改坐标轴标签的文本？

方法

使用 xlab() 或 ylab() 来修改坐标轴标签的文本（见图 8-19）：

```
library(gcookbook) # 为了使用数据集

hwp <- ggplot(heightweight, aes(x=ageYear, y=heightIn, colour=sex)) +
       geom_point()
# 使用默认的坐标轴标签
hwp

# 设置坐标轴标签
hwp + xlab("Age in years") + ylab("Height in inches")
```

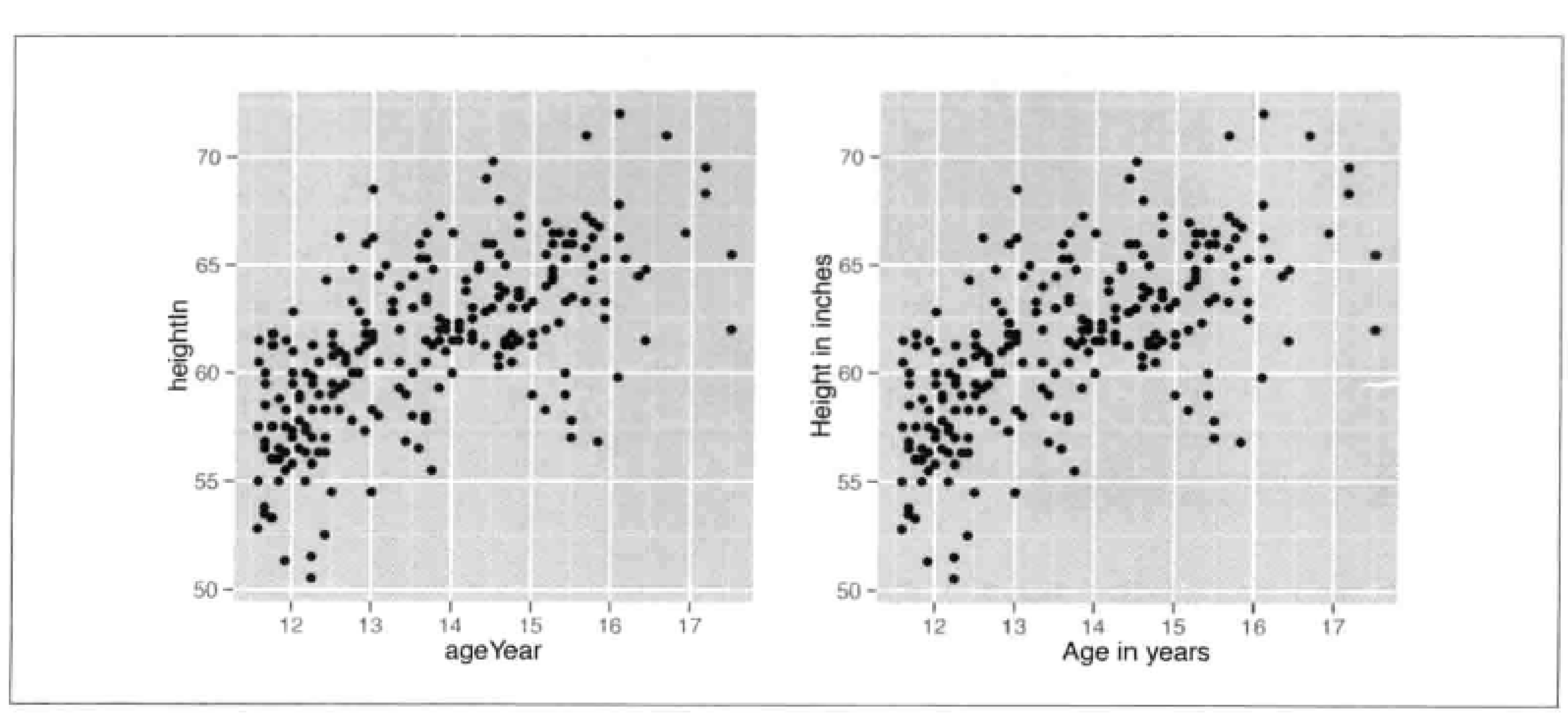


图 8-19　左图：使用默认坐标轴标签的散点图　右图：为 x 轴和 y 轴手动指定了坐标轴标签的散点图

讨论

默认情况下，图形将直接使用数据框中的列名作为坐标轴标签。这对于探索数据来说可能还好，但是在对外呈现数据时，你也许会希望使用更具描述力的坐标轴标签。

除了 xlab() 和 ylab()，也可以使用 labs()：

```
hwp + labs(x = "Age in years", y = "Height in inches")
```

设置坐标轴标签的另一种方法是在标度中指定，就像这样：

```
hwp + scale_x_continuous(name="Age in years")
```

这种方法看起来可能有点别扭，不过可能在你同时设定标度的其他属性（如刻度线位置、值域等）时会比较有用。

当然，这种方法同样适用于其他的坐标轴标度，如 scale_y_continuous()、scale_x_discrete() 等。

还可以使用 \n 来添加换行，如图 8-20 所示：

```
hwp + scale_x_continuous(name="Age\n(years)")
```

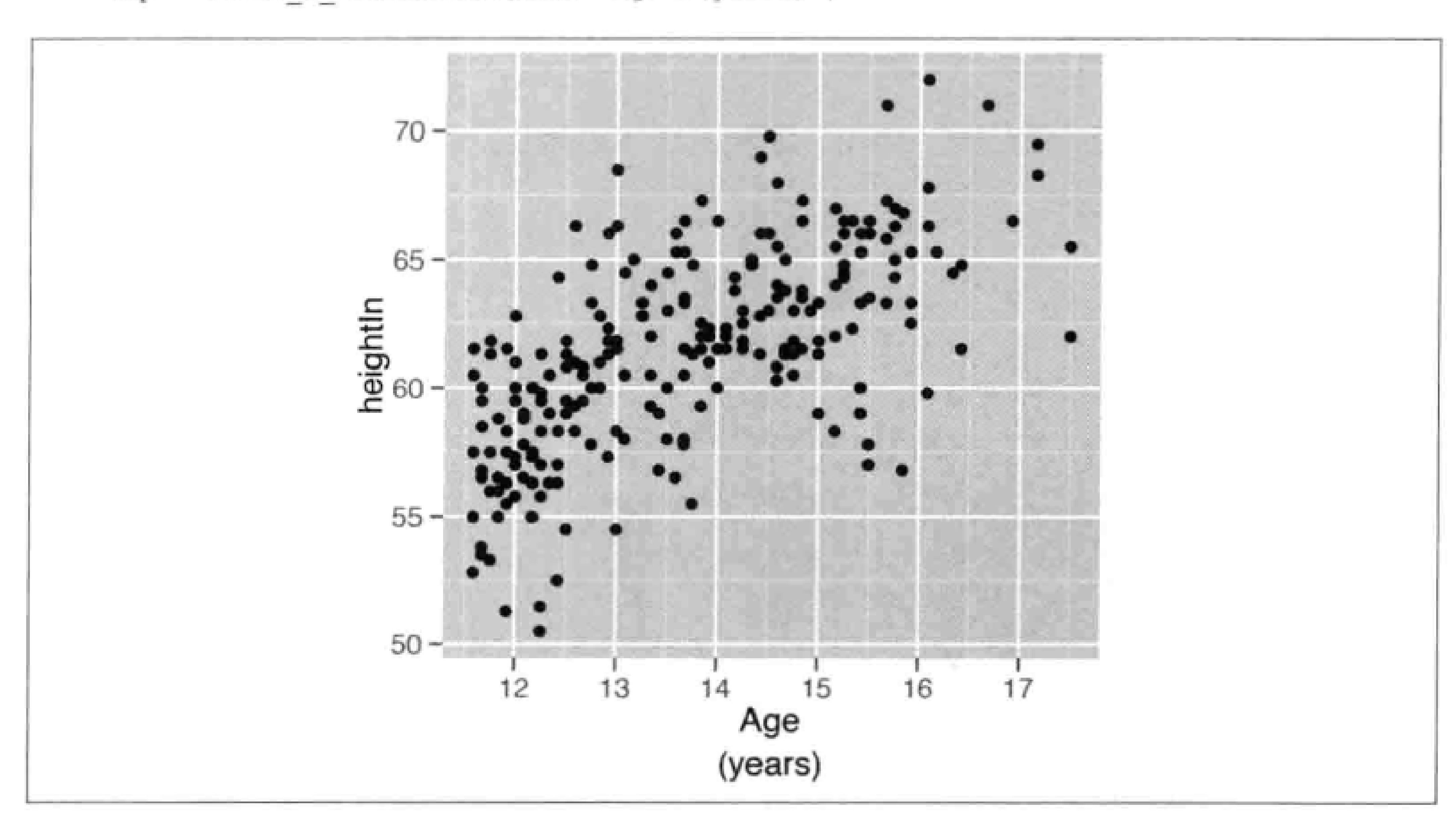


图 8-20 含一个换行的 x 轴标签

8.11 移除坐标轴标签

问题

如何移除某条坐标轴的标签？

方法

对于 *x* 轴标签，使用 theme(axis.title.x=element_blank())。对于 *y* 轴标签，针对 axis.title.y 做同样处理。

我们将在本例中隐藏 *x* 轴标签（见图 8-21）：

```
p <- ggplot(PlantGrowth, aes(x=group, y=weight)) + geom_boxplot()

p + theme(axis.title.x=element_blank())
```

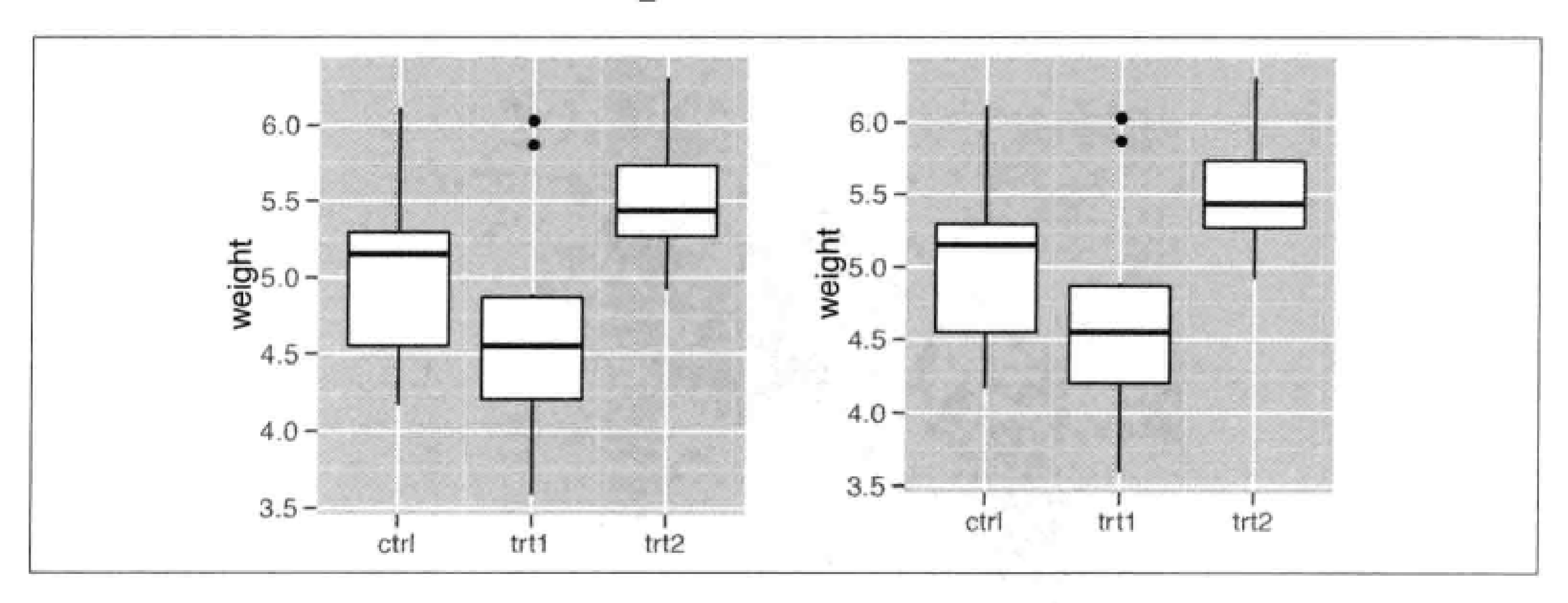


图 8-21　左图：使用 **element_blank()** 处理的 *x* 轴标签　右图：将标签设为 ""

讨论

某些坐标轴标签对于上下文来说是冗余的或者是显而易见的，因此并不需要显示。在这个示例中，*x* 轴表示变量 group，但是这一点可以很明显地从上下文看出来。类似地，如果 *y* 轴在每个刻度标签上都标注 kg（千克）或类似的单位，则坐标轴标签 “weight” 就没有必要显示了。

移除坐标轴标签的另一种方法是将其设为一个空字符串。但如果以这种方式去做，那么图中将仍为文本留出空间，如图 8-21 右图所示：

```
p + xlab("")
```

当你使用 theme() 来设置 axis.title.x=element_blank() 时，*x* 或 *y* 标度的名称是不会改变的，只是这样不会显示文本而且不会为其留出空间。当你设置标签为 "" 时，标度的名称就改变了，并且实际上显示了（空白的）文本。

8.12　修改坐标轴标签的外观

问题

如何修改坐标轴标签的外观？

方法

要修改 x 轴标签的外观（见图 8-22），使用 axis.title.x 即可：

```
library(gcookbook) # 为了使用数据集

hwp <- ggplot(heightweight, aes(x=ageYear, y=heightIn)) + geom_point()

hwp + theme(axis.title.x=element_text(face="italic", colour="darkred", size=14))
```

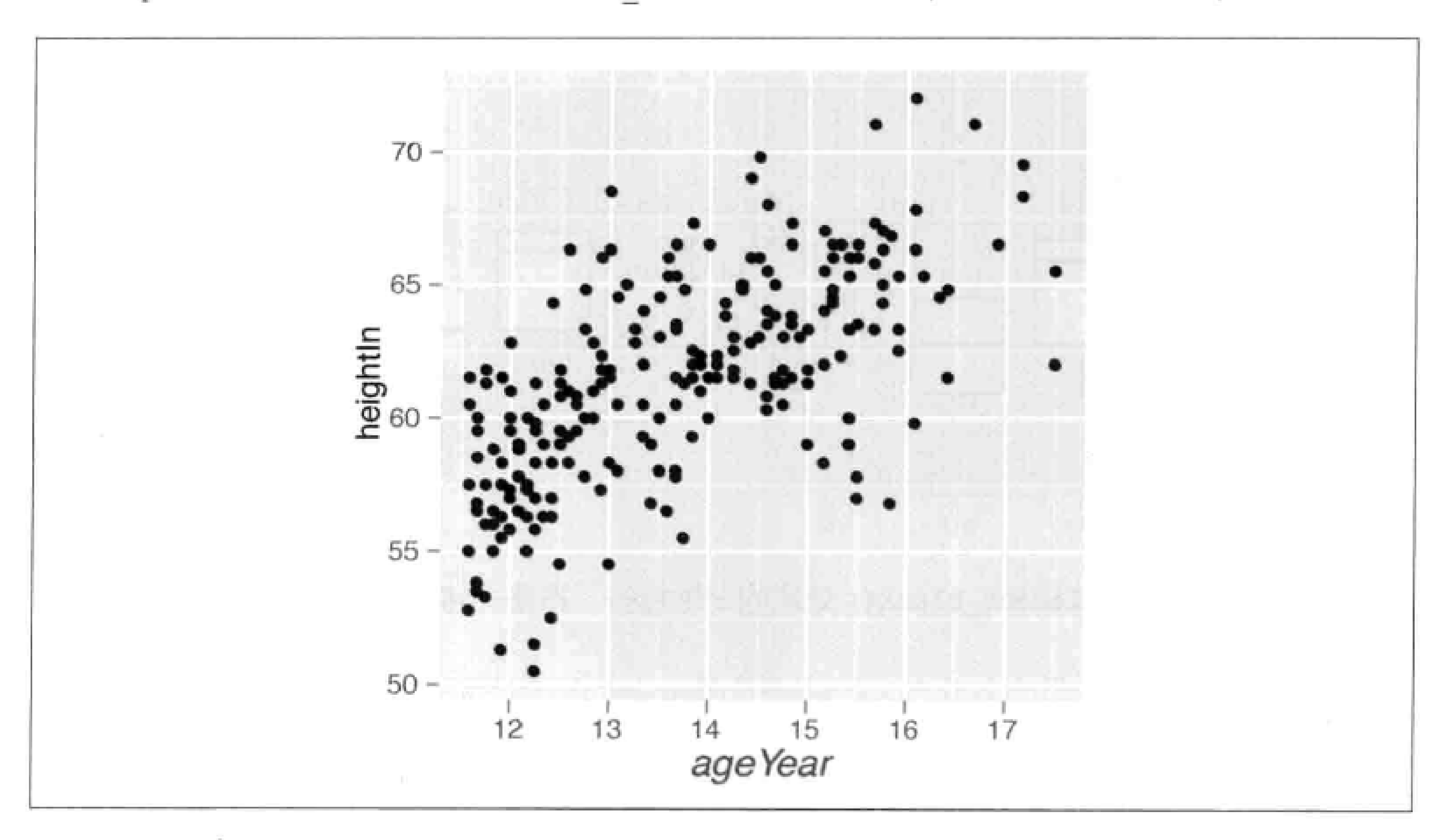


图 8-22　自定义外观的 x 轴标签

讨论

对于 y 轴标签来说，有时不对文本进行旋转会比较有用，如图 8-23 左图所示。标签中的 \n 表示另起一行：

```
hwp + ylab("Height\n(inches)") +
    theme(axis.title.y=element_text(angle=0, face="italic", size=14))
```

当调用 element_text() 时，默认的角度是 0，所以如果设置了 axis.title.y 但没有指定这个角度，它将以文本的顶部指向上方的朝向显示。如果修改了 axis.title.y 中的其他任何属性并且希望它以正常朝向，即旋转 90° 显示，则必须手动指定这个角度（见图 8-23 右图）：

```
hwp + ylab("Height\n(inches)") +
    theme(axis.title.y=element_text(angle=90, face="italic", colour="darkred",
                                    size=14))
```

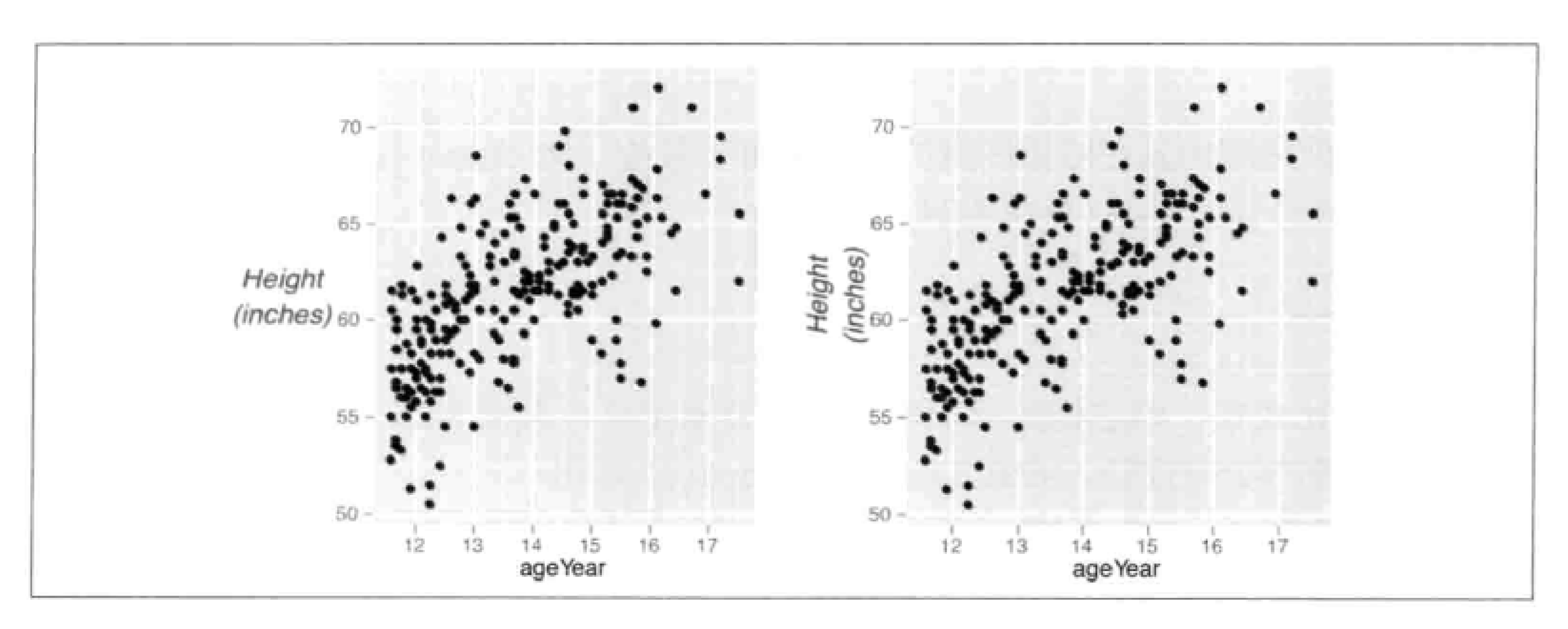


图 8-23　左图：angle=0 的 y 轴标签　右图：angle=90 的 y 轴标签

另见

参见 9.2 节以了解更多关于如何控制文本外观的信息。

8.13　沿坐标轴显示直线

问题

如何沿 x 轴和 y 轴显示直线，但不在图形的另两侧显示？

方法

使用主题设置中的 axis.line（见图 8-24）：

```
library(gcookbook) # 为了使用数据集

p <- ggplot(heightweight, aes(x=ageYear, y=heightIn)) + geom_point()

p + theme(axis.line = element_line(colour="black"))
```

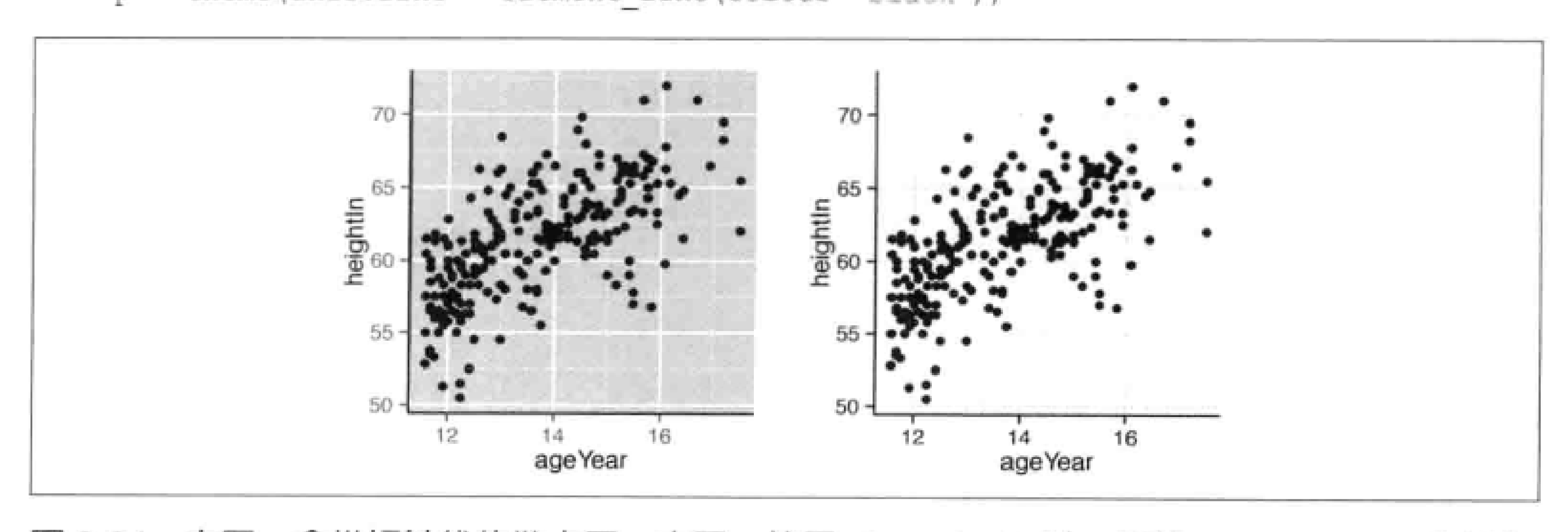


图 8-24　左图：含坐标轴线的散点图　右图：使用 theme_bw() 时，需将 panel.border 也设为空白

讨论

如果你最初使用的主题在绘图区域的周围就有一条边（如 theme_bw()），则需要同时重置参数 panel.border（见图 8-24 右图）：

```
p + theme_bw() +
    theme(panel.border = element_blank(),
          axis.line = element_line(colour="black"))
```

如果边界线比较粗，则它们的末端将仅会部分地重叠（见图 8-25 左图）。要让它们完全重叠（见图 8-25 右图），设置 lineend="square" 即可：

```
# 对于较粗的线条，只有一半重叠
p + theme_bw() +
    theme(panel.border = element_blank(),
          axis.line = element_line(colour="black", size=4))

# 完全重叠
p + theme_bw() +
    theme(panel.border = element_blank(),
          axis.line = element_line(colour="black", size=4, lineend="square"))
```

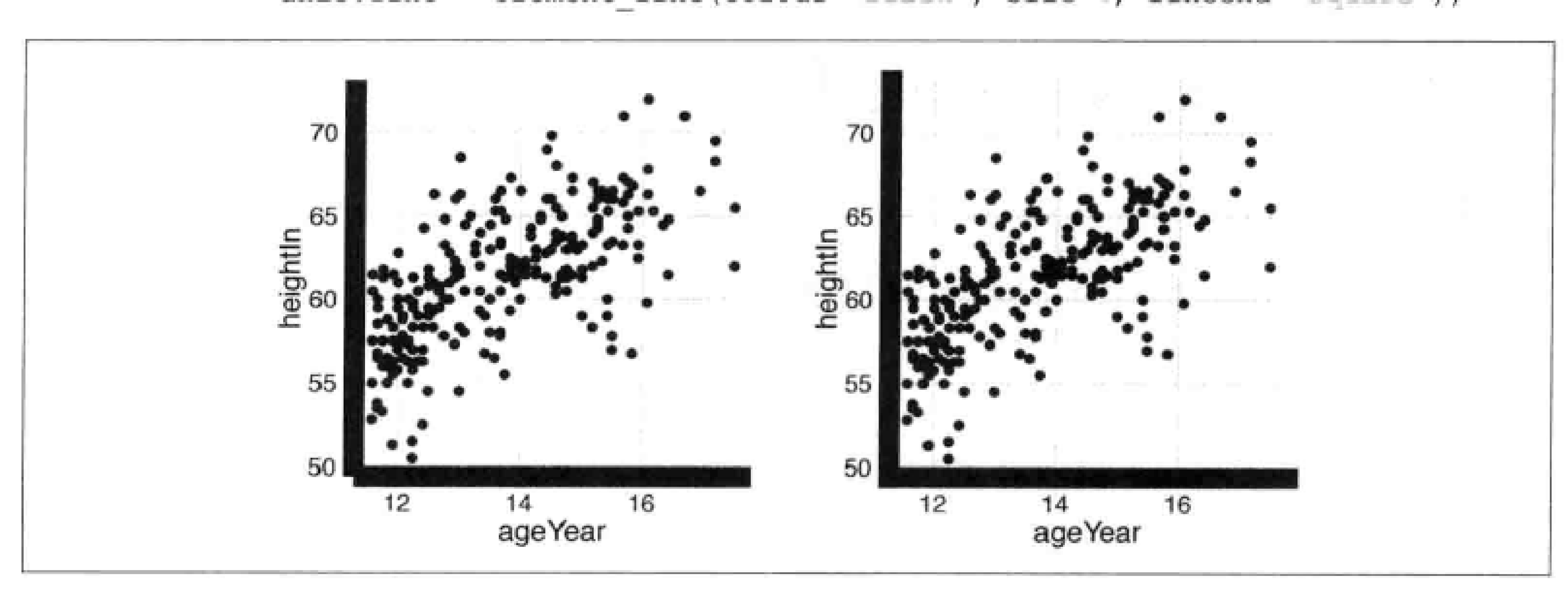


图 8-25　左图：对于粗线条，末端不会完全重叠　右图：使用 lineend="square" 令其完全重叠

另见

关于主题系统工作原理的更多信息，参见 9.3 节。

8.14　使用对数坐标轴

问题

如何在一幅图上使用对数坐标轴？

方法

使用 scale_x_log10() 和 / 或 scale_y_log10()（见图 8-26）：

```
library(MASS) # 为了使用数据集

# 基本图形
p <- ggplot(Animals, aes(x=body, y=brain, label=rownames(Animals))) +
    geom_text(size=3)
p

# 使用对数 x 标度和对数 y 标度
p + scale_x_log10() + scale_y_log10()
```

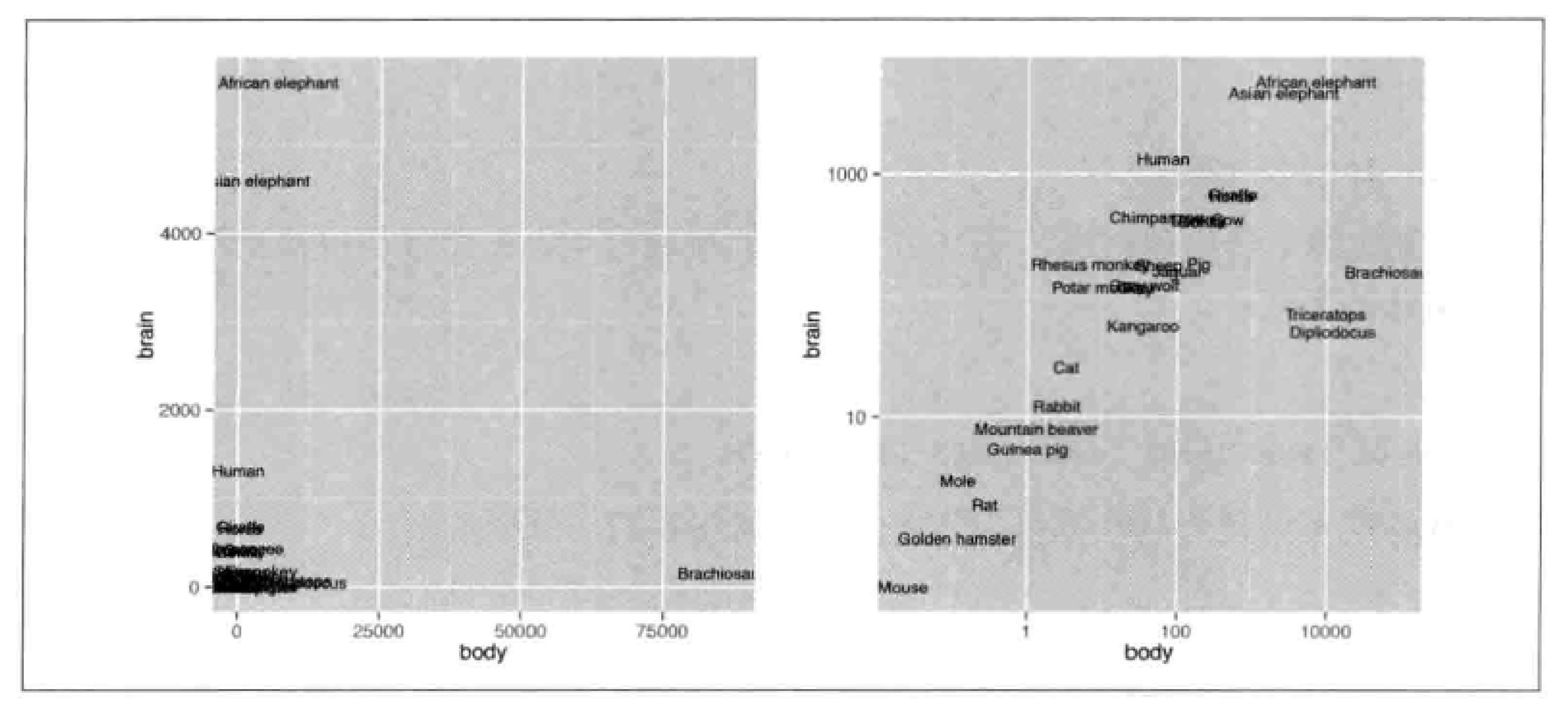


图 8-26　左图：使用线性标度坐标轴展示呈指数分布的数据　右图：使用对数坐标轴

讨论

使用对数坐标轴时，视觉上某段给定的距离表示着常数倍的比例改变；举例来说，*y* 轴上每增加 1 厘米可能表示数量乘以 10。相对而言，使用线性坐标轴时，视觉上某段给定的距离表示着常数单位数量的改变；每增加 1 厘米可能表示数量上增加了 10。

某些数据集在 *x* 轴上是呈指数分布的，而另一些则是在 *y* 轴上呈指数分布（或者两轴皆是）。例如，MASS 库中的 Animals 数据集包含了各类哺乳动物平均脑质量（单位为 g）和体重（单位为 kg）数据，还加入了若干种恐龙的数据作为对照：

```
Animals

                      body  brain
Mountain beaver      1.350    8.1
Cow                465.000  423.0
Grey wolf           36.330  119.5
...
Brachiosaurus    87000.000  154.5
Mole                 0.122    3.0
Pig                192.000  180.0
```

如图 8-26 所示，我们可以绘制一幅散点图来对脑质量和体重之间的关系进行可视化。

在使用默认的线性标度坐标轴时，我们很难更好地理解这幅图。由于几种大型动物的存在，其余的动物都被挤到了左下角——这让老鼠（mouse）与三角龙（triceratops）看起来几乎没有区别！这就是一个数据在两条坐标轴上均呈指数分布的例子。

关于将刻度线放到何处的问题，ggplot2 会试着做出明智的选择，但是如果你不喜欢这些刻度，那么可以通过指定 `breaks`（也可再额外指定 `labels`）来修改它们。在这个示例中，自动生成刻度线的间距较理想的间距更远。针对 y 轴的刻度线，我们可以像下面这样获得一个含有从 10^0 到 10^3 的 10 的各次幂的向量：

```
10^(0:3)
```

```
1   10  100 1000
```

x 轴刻度线的工作原理相同，不过由于这里的值域过大，R 会自动将输出格式化为科学记数法的形式：

```
10^(-1:5)
```

```
1e-01 1e+00 1e+01 1e+02 1e+03 1e+04 1e+05
```

之后我们就可以使用这些值作为分割点了，如图 8-27 左图所示：

```
p + scale_x_log10(breaks=10^(-1:5)) + scale_y_log10(breaks=10^(0:3))
```

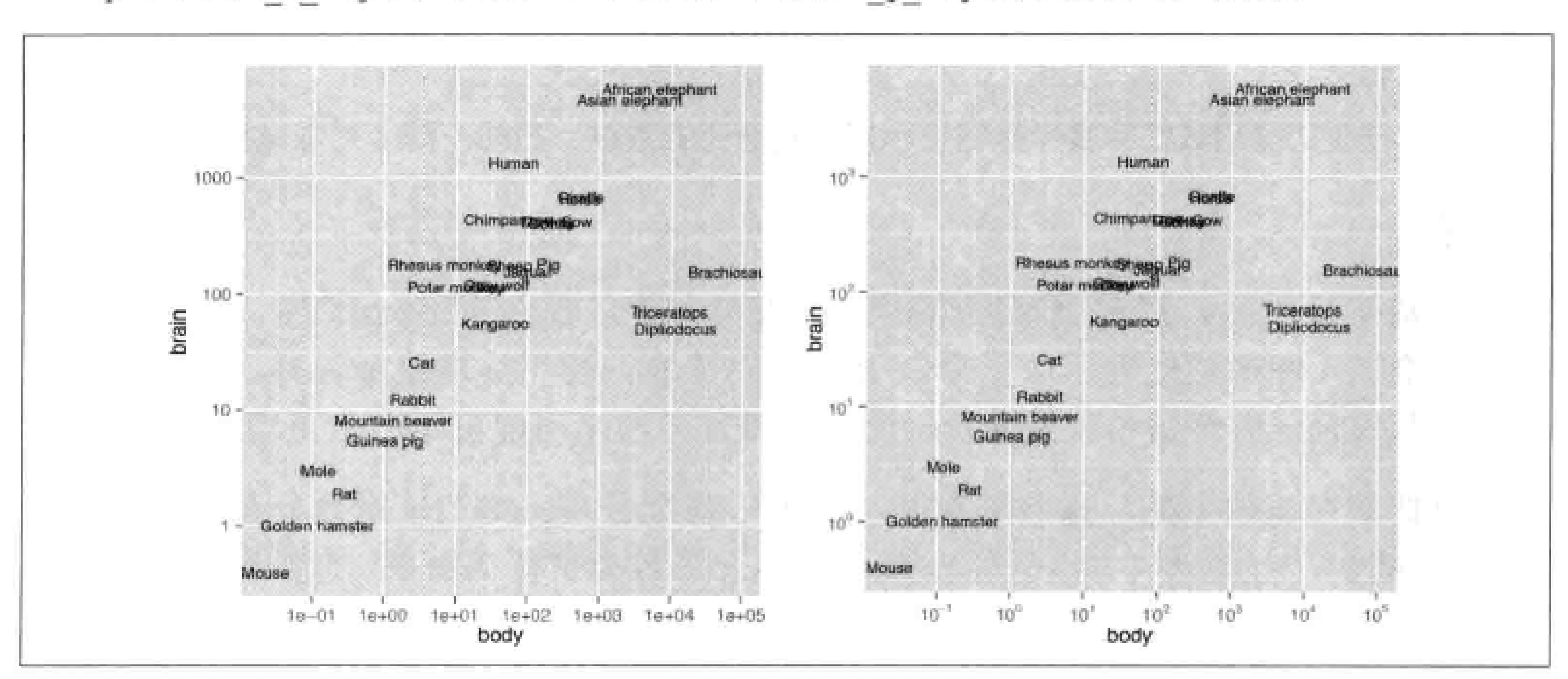


图 8-27　左图：x 轴和 y 轴取以 10 为底对数的散点图，并手动指定了刻度的位置　右图：指数表示的刻度标签

要让刻度标签转而使用指数记数法（见图 8-27 右图），只要使用 `scales` 包中的函数 `trans_format()` 即可：

```
library(scales)
p + scale_x_log10(breaks=10^(-1:5),
                  labels=trans_format("log10", math_format(10^.x))) +
    scale_y_log10(breaks=10^(0:3),
                  labels=trans_format("log10", math_format(10^.x)))
```

使用对数坐标轴的另一种方法是，在将数据映射到 x 和 y 坐标之前，先对其进行变换（见图 8-28）。从技术上讲，坐标轴仍然是线性的——它表示对数变换后的数值：

```
ggplot(Animals, aes(x=log10(body), y=log10(brain), label=rownames(Animals))) +
    geom_text(size=3)
```

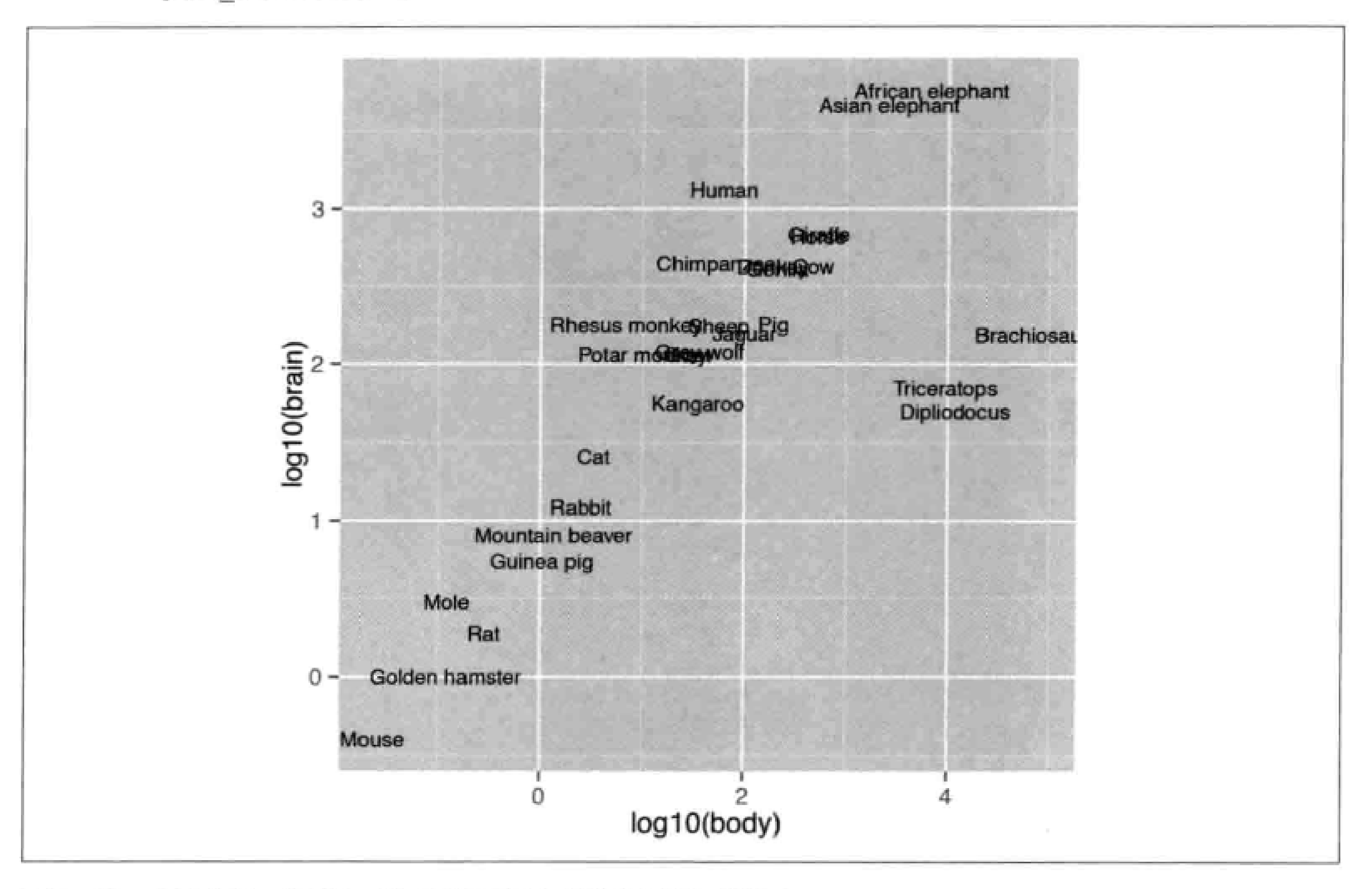


图 8-28　映射到 x 轴和 y 轴之前先进行对数变换再绘图

上例中仅使用了一个 $\log_{10}$ 变换，不过使用其他的变换也是可以的，如以 2 为底的对数变换和自然对数变换，如图 8-29 所示。使用这些变换有点复杂——`scale_x_log10()` 可以简写，但是对于其他的对数标度而言，我们需要完整地定义它们：

```
library(scales)

# 对 x 使用自然对数变换，对 y 使用 log2 变换
p + scale_x_continuous(trans = log_trans(),
                       breaks = trans_breaks("log", function(x) exp(x)),
                       labels = trans_format("log", math_format(e^.x))) +
    scale_y_continuous(trans = log2_trans(),
                       breaks = trans_breaks("log2", function(x) 2^x),
                       labels = trans_format("log2", math_format(2^.x)))
```

我们也可以只使用一条对数坐标轴。这种做法对于呈现金融数据往往是有用的，因为这样能够更好地展示出按比例的变化。图 8-30 分别使用了线性和对数的 y 轴来展示苹果公司的股价变化情况。默认的刻度线间距对你的图来说可能并不够好；可以在标度中使用参数 `breaks` 来设置它们：

```
library(gcookbook) # 为了使用数据集

ggplot(aapl, aes(x=date,y=adj_price)) + geom_line()

ggplot(aapl, aes(x=date,y=adj_price)) + geom_line() +
    scale_y_log10(breaks=c(2,10,50,250))
```

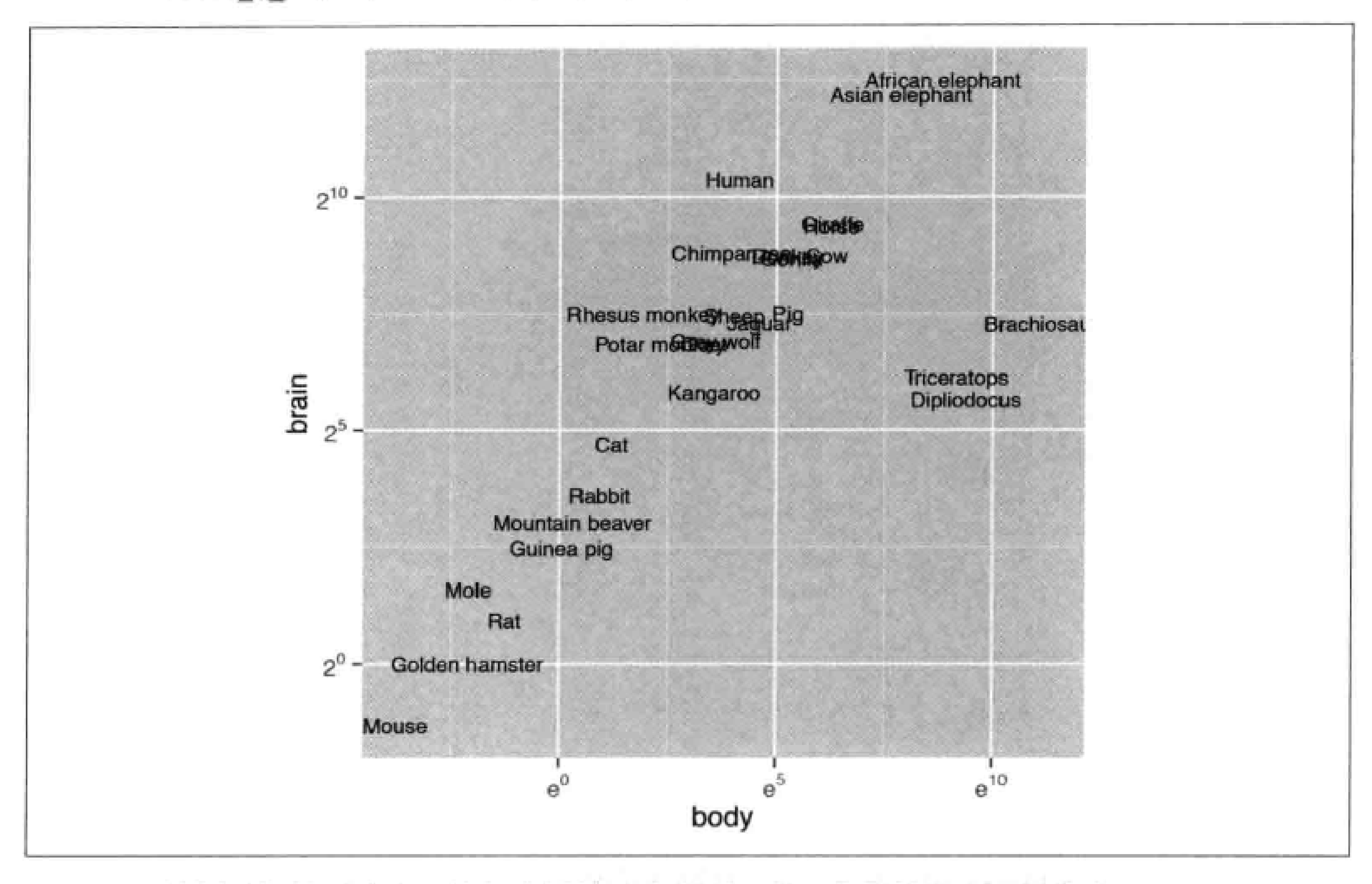


图 8-29　绘制以指数形式表示的刻度标签。注意对 *x* 和 *y* 分别使用了不同的底

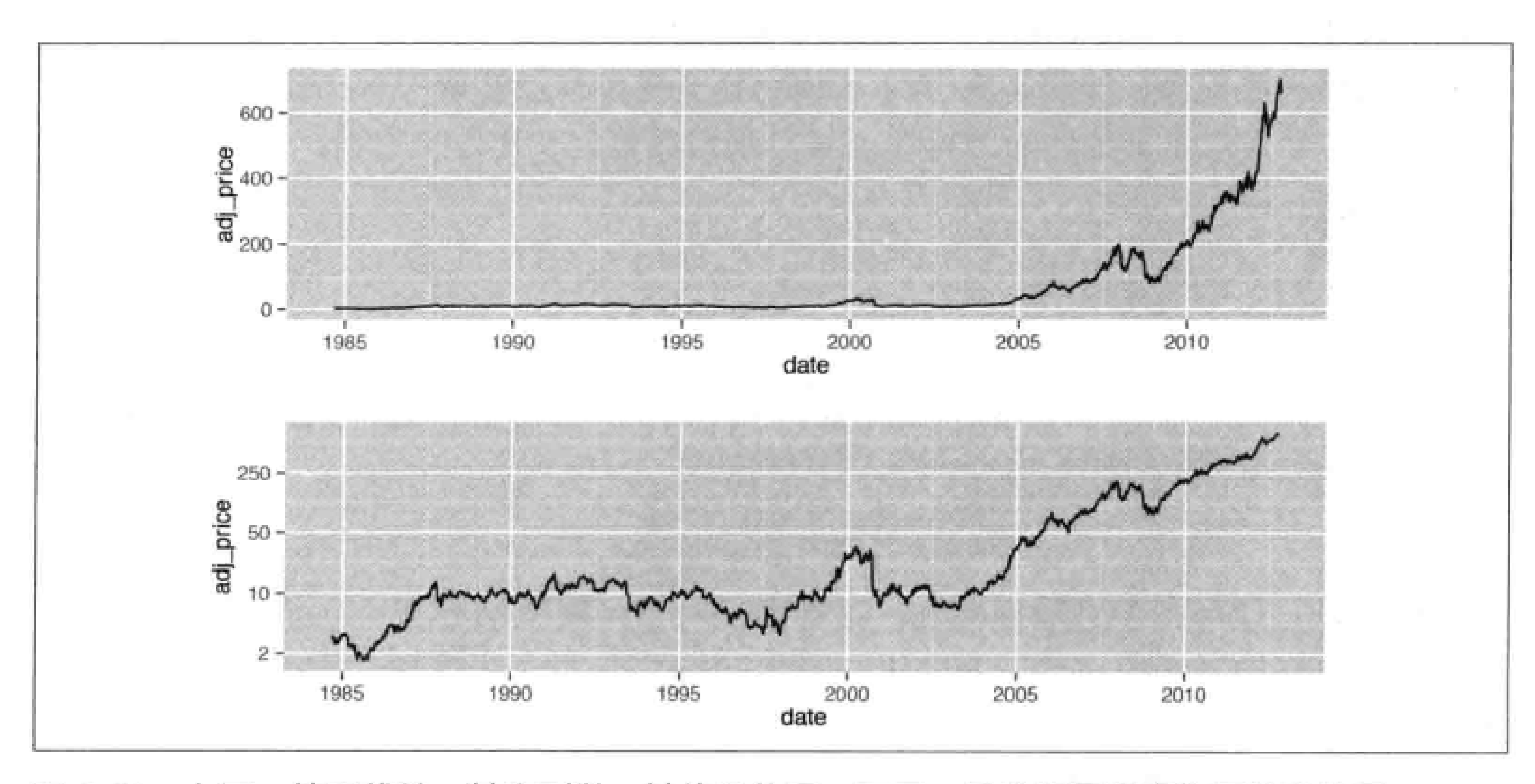


图 8-30　上图：使用线性 *x* 轴和对数 *y* 轴的股价图　下图：手动设置刻度位置的股价图

8.15 为对数坐标轴添加刻度

问题

如何为对数坐标轴添加间距递减的刻度线？

方法

使用 annotation_logticks()（见图 8-31）：

```
library(MASS)   # 为了使用数据集
library(scales) # 为了使用 trans 和 format 相关函数
ggplot(Animals, aes(x=body, y=brain, label=rownames(Animals))) +
    geom_text(size=3) +
    annotation_logticks() +
    scale_x_log10(breaks = trans_breaks("log10", function(x) 10^x),
                  labels = trans_format("log10", math_format(10^.x))) +
    scale_y_log10(breaks = trans_breaks("log10", function(x) 10^x),
                  labels = trans_format("log10", math_format(10^.x)))
```

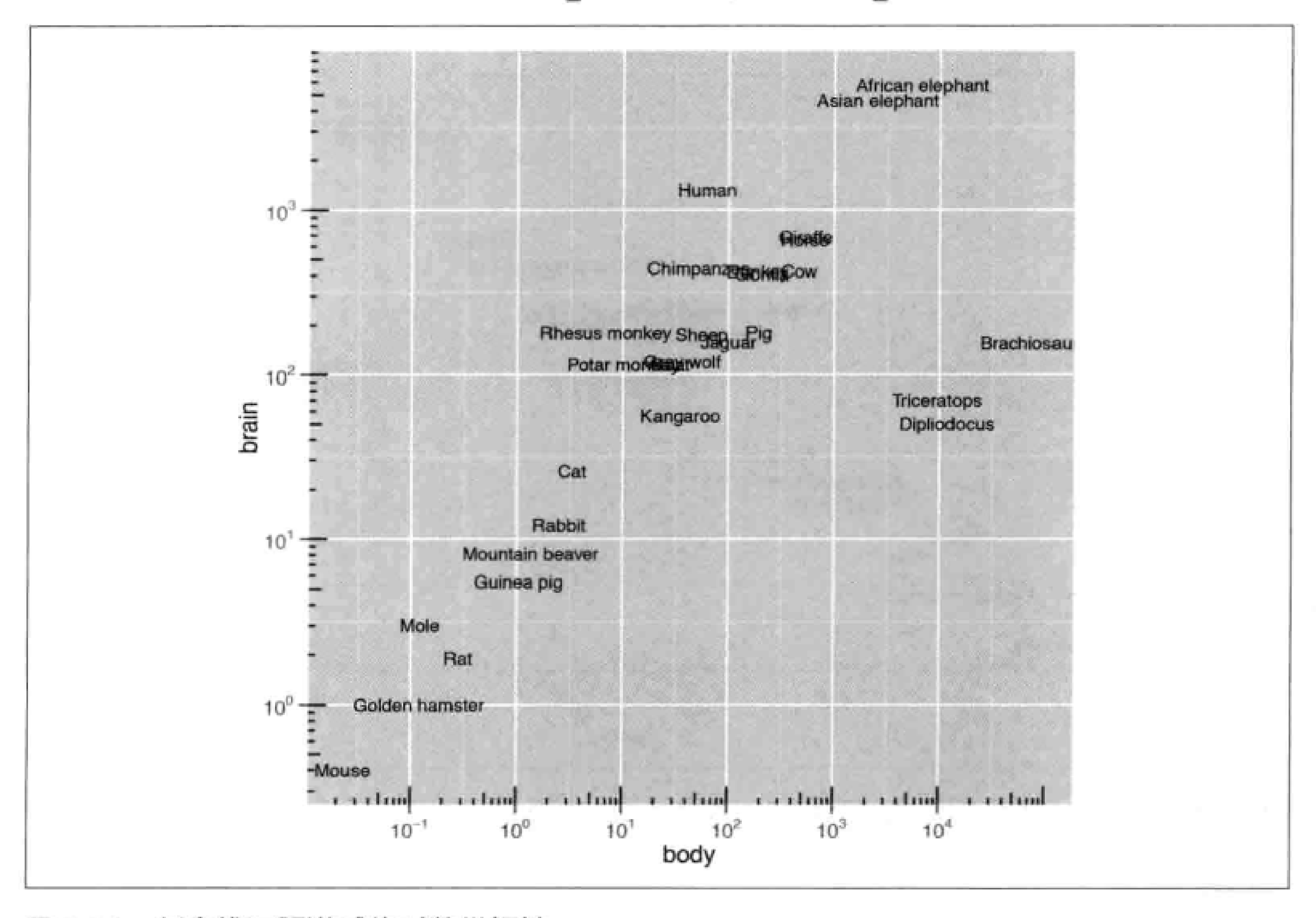


图 8-31 刻度线间距递减的对数坐标轴

讨论

使用 annotation_logticks() 创建的刻度线事实上是绘图区域中的几何对象。在每个

10 的幂次处有一条长刻度线，在每个 5 的位置处有一条中等长度的刻度线①。

你可以使用 theme_bw() 让刻度线和网格线的颜色更协调一些。

默认情况下，次网格线在视觉上出现在两条主网格线的正中间，但这与对数标度下表示“5”的刻度线位置并不相同。要让两者位置相同，可以手动设定标度的 minor_breaks 参数。要完成这里的任务，我们需要将它们设置为 log10(5*10^(minpow:maxpow))，也可以缩写为 log10(5) + minpow:maxpow（见图 8-32）：

```
ggplot(Animals, aes(x=body, y=brain, label=rownames(Animals))) +
    geom_text(size=3) +
    annotation_logticks() +
    scale_x_log10(breaks = trans_breaks("log10", function(x) 10^x),
                  labels = trans_format("log10", math_format(10^.x)),
                  minor_breaks = log10(5) + -2:5) +
    scale_y_log10(breaks = trans_breaks("log10", function(x) 10^x),
                  labels = trans_format("log10", math_format(10^.x)),
                  minor_breaks = log10(5) + -1:3) +
    coord_fixed() +
    theme_bw()
```

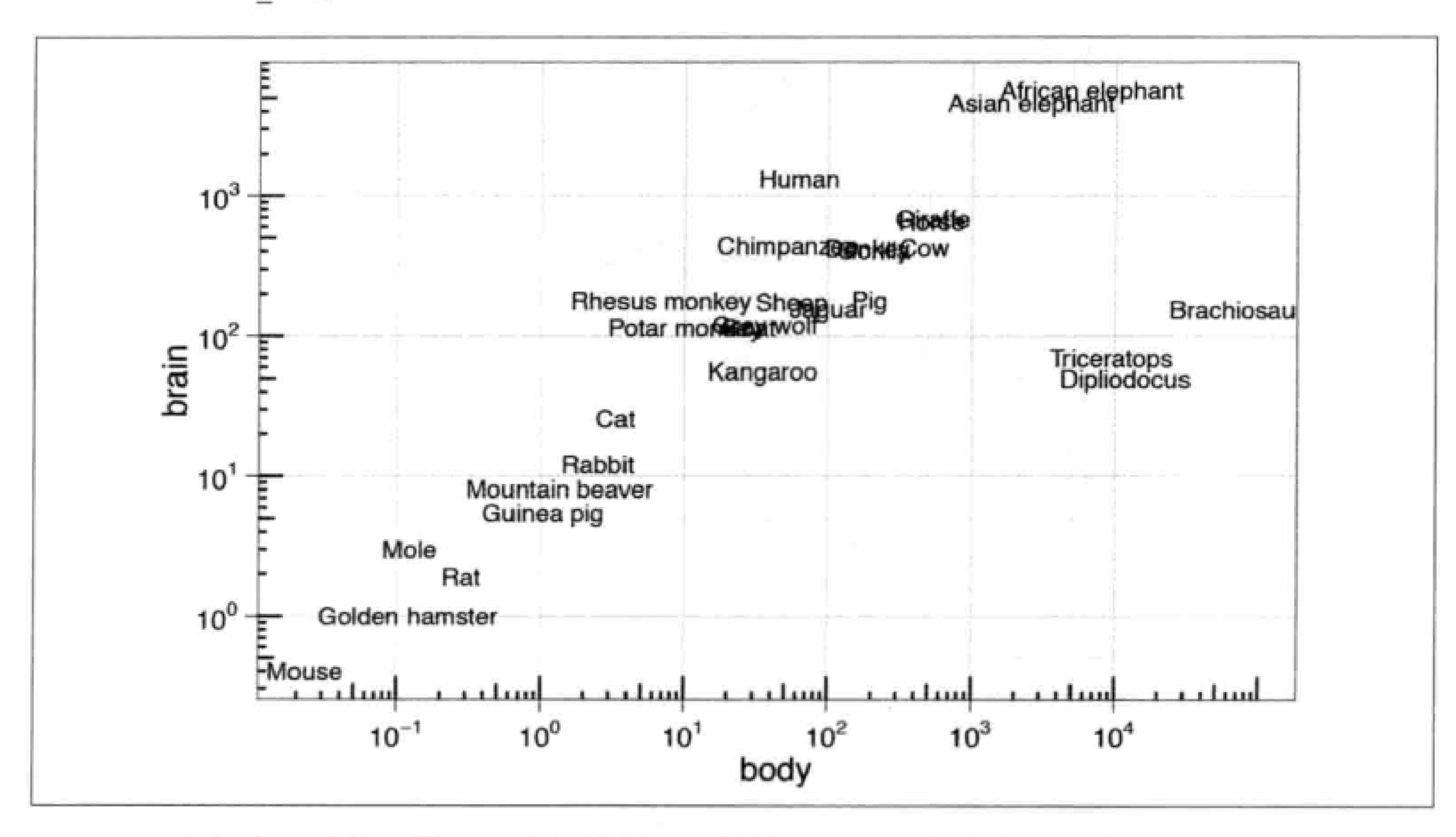


图 8-32　在每个 5 的位置带有刻度线的对数坐标轴，以及固定的坐标比例

另见

关于控制 x 轴和 y 轴缩放比例的更多知识，参见 8.5 节。

① 这里所谓 5 的位置，是指前一个长刻度线对应数值 5 倍的位置。——译者注

8.16 绘制环状图形

问题

如何绘制一幅环状图形①？

方法

使用 `coord_polar()`。对于本例，我们将使用 `gcookbook` 包中的 `wind` 数据集。它包含了某一天中每隔 5 分钟的风速和风向样本。风向每隔 15° 被分到一个组中，风速则按每 5 m/s 分为子样本：

```
library(gcookbook) # 为了使用数据集
wind
```

```
 TimeUTC  Temp WindAvg WindMax WindDir SpeedCat DirCat
       0  3.54    9.52   10.39      89    10-15     90
       5  3.52    9.10    9.90      92     5-10     90
      10  3.53    8.73    9.51      92     5-10     90
 ...
    2335  6.74   18.98   23.81     250      >20    255
    2340  6.62   17.68   22.05     252      >20    255
    2345  6.22   18.54   23.91     259      >20    255
```

我们将使用 `geom_histogram()` 对每个 `SpeedCat` 和 `DirCat` 的类别绘制样本数量的计数值（见图 8-33）。我们将 `binwidth` 设置为 15 以使直方图的 `origin` 开始于 -7.5 的位置，这样每个扇形就会居中于 0、15、30 等位置：

```
ggplot(wind, aes(x=DirCat, fill=SpeedCat)) +
    geom_histogram(binwidth=15, origin=-7.5) +
    coord_polar() +
    scale_x_continuous(limits=c(0,360))
```

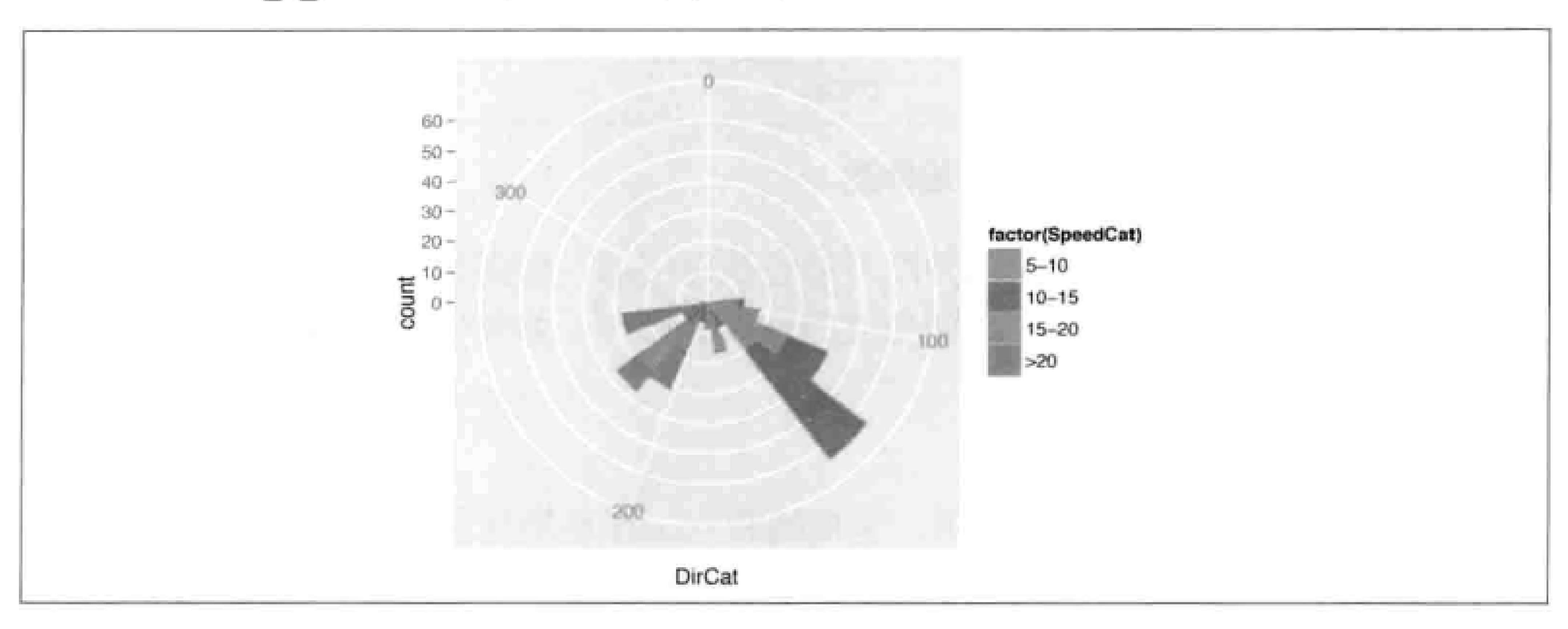


图 8-33 极坐标图

① circular graph，在这里作者特指极坐标图。——译者注

讨论

使用极坐标图时要小心，因为这种图形会扭曲对数据的感知。本例中，在 210° 的位置有 15 个风速为 15-20 的观测以及 13 个风速大于 20 的观测，但是对图形匆匆一瞥时，看起来好像风速大于 20 的观测更多一些。而且还存在三个风速 10-15 的观测，它们却几乎不可见。

在这个例子中，我们可以通过反转图例、使用不同的调色板、添加外框线以及将分割点设置为某些更熟悉的值的方式，让图形稍微美观一些（见图 8-34）：

```
ggplot(wind, aes(x=DirCat, fill=SpeedCat)) +
    geom_histogram(binwidth=15, origin=-7.5, colour="black", size=.25) +
    guides(fill=guide_legend(reverse=TRUE)) +
    coord_polar() +
    scale_x_continuous(limits=c(0,360), breaks=seq(0, 360, by=45),
                       minor_breaks=seq(0, 360, by=15)) +
    scale_fill_brewer()
```

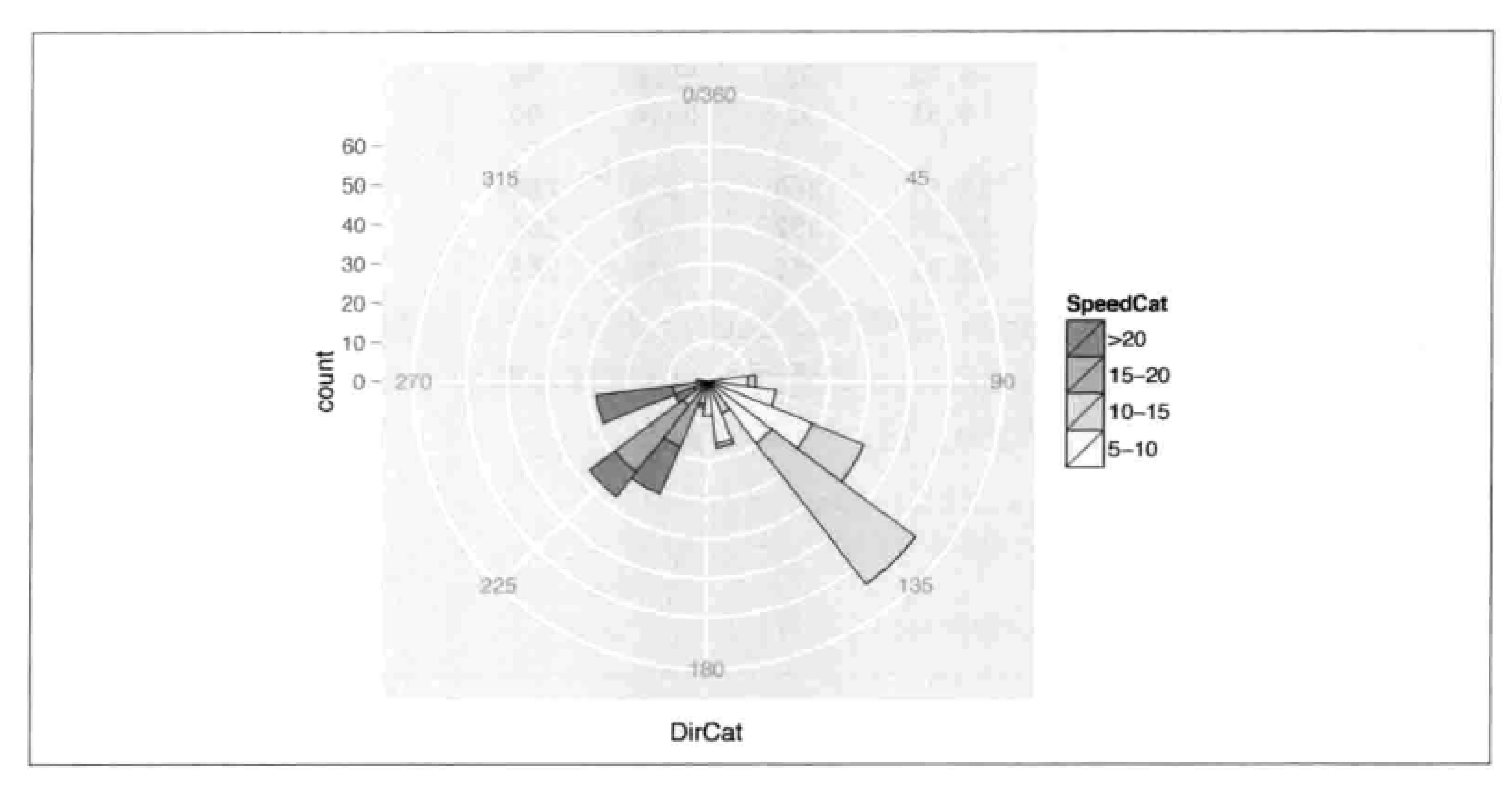


图 8-34　使用不同颜色和分割点的极坐标图

使用参数 `start` 设置图形起始的角度可能也是有用的，特别是当我们使用一个离散型变量映射为角度（theta）时[①]。起始角度的值以弧度计，如果你知道要调整的角度，则必须将它转换为弧度：

```
coord_polar(start=-45 * pi / 180)
```

极坐标可与其他几何对象搭配使用，包括线和点。在使用这些几何对象时有一些重要的问题要牢记于心。首先，默认情况下，对于映射到 y（或者说 r）的变量，最小值将被映射到中心；换句话说，数据中的最小值将被映射到视觉上半径为 0 的位置。你可

① 这里的极坐标系与通常的定义相同，为 (r, θ)，其中 r 为半径，θ 为角度。——译者注

能希望一个为 0 的数据值被映射到半径为 0 的位置，但是为了确保图形能这样绘制，需要设置对应的界限（limit）。

下一个问题是，在使用一个连续型的 *x*（或者说 theta）时，数据中的最小值和最大值是重合的。有时这样是可取的，有时却不是。要修改这种默认行为，你需要设置对应的界限。

最后，极坐标的 theta 值不能环绕一周——目前还无法制作一个穿越过起始角度（通常为垂直方向）的几何对象。

我们将使用一个示例来阐明这些问题。以下代码根据时间序列数据集 `mdeaths` 创建了一个数据框并绘制了图 8-35 左侧的图形：

```
# 将 mdeaths 的时间序列数据放入一个数据框
md <- data.frame(deaths = as.numeric(mdeaths),
                 month = as.numeric(cycle(mdeaths)))

# 计算每个月的平均死亡数量
library(plyr) # 为了使用 ddply() 函数
md <- ddply(md, "month", summarise, deaths = mean(deaths))
md

 month   deaths
     1 2129.833
     2 2081.333
 ...
    11 1377.667
    12 1796.500

#绘制基本图形
p <- ggplot(md, aes(x=month, y=deaths)) + geom_line() +
    scale_x_continuous(breaks=1:12)

# 使用 coord_polar
p + coord_polar()
```

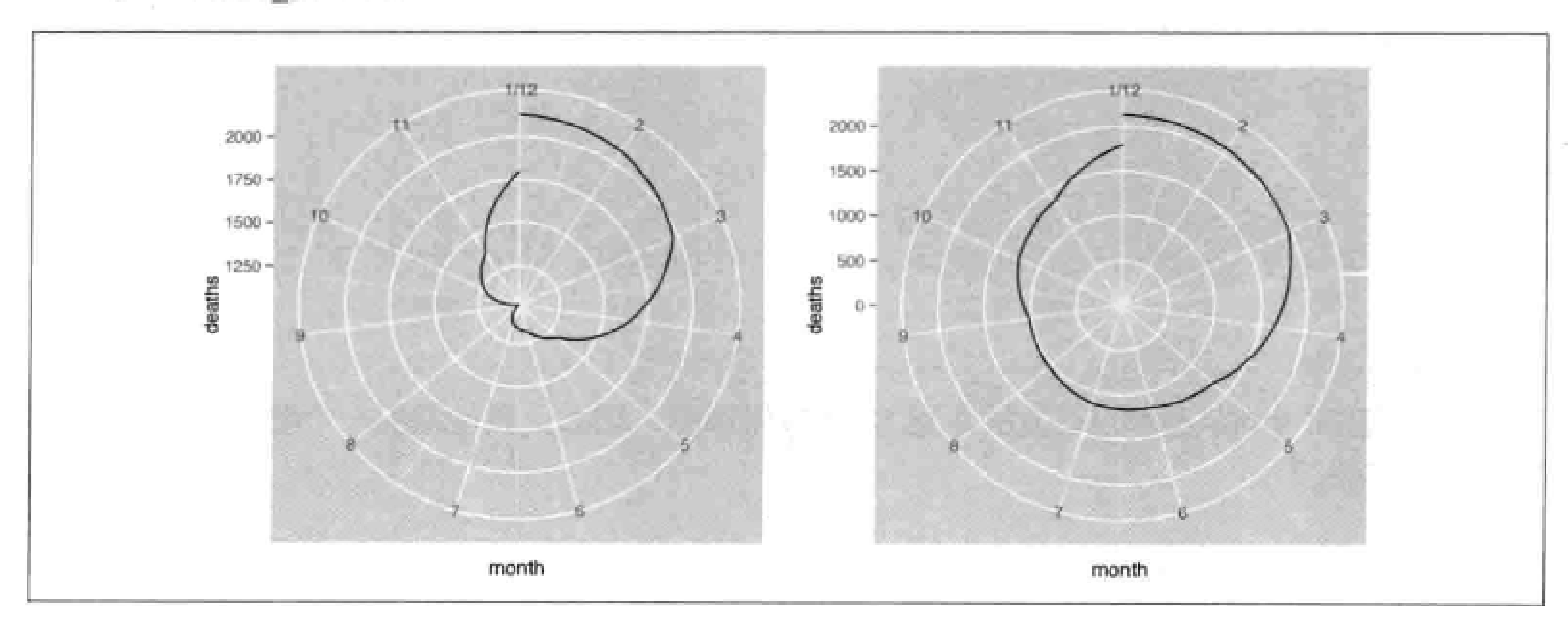


图 8-35　左图：使用线几何对象的极坐标图（注意半径代表的数据范围）　右图：半径表示的数据范围从 0 开始

第一个问题是，数据的值（范围大约是 1000 ～ 2100）被映射为半径，于是最小的数据值就处于半径为 0 的位置。我们将通过设置 y（或者说 r）的界限为从 0 到数据中的最大值来解决这个问题，如图 8-35 右图所示：

```
# 使用 coord_polar 并将 y (r) 的下界设置为 0
p + coord_polar() + ylim(0, max(md$deaths))
```

下一个问题是最小和最大的 month（月份）值 1 和 12 被展示在了同样的角度上。我们将通过设置 x 的界限为 0 ～ 12 来解决这个问题，绘制的图形为图 8-36 左图（注意 xlim() 的使用覆盖了 p 中的 scale_x_continuous()，所以它将不再为每个月份显示分割点；参见 8.2 节以了解更多信息）：

```
p + coord_polar() + ylim(0, max(md$deaths)) + xlim(0, 12)
```

还有最后一个首尾不相接的问题。要解决这个问题，我们需要修改数据框，添加一个月份为 0，对应值与 12 月相同的行。这将使得起点和终点变得相同，如图 8-36 右图所示（或者，我们可以添加一个 13 月，而非 0 月）：

```
# 通过添加一个值与 12 的值相同的 0 来连接曲线
mdx <- md[md$month==12, ]
mdx$month <- 0
mdnew <- rbind(mdx, md)

# 通过使用 %+% ，绘制与之前相同的图形，只是使用的数据不同
p %+% mdnew + coord_polar() + ylim(0, max(md$deaths))
```

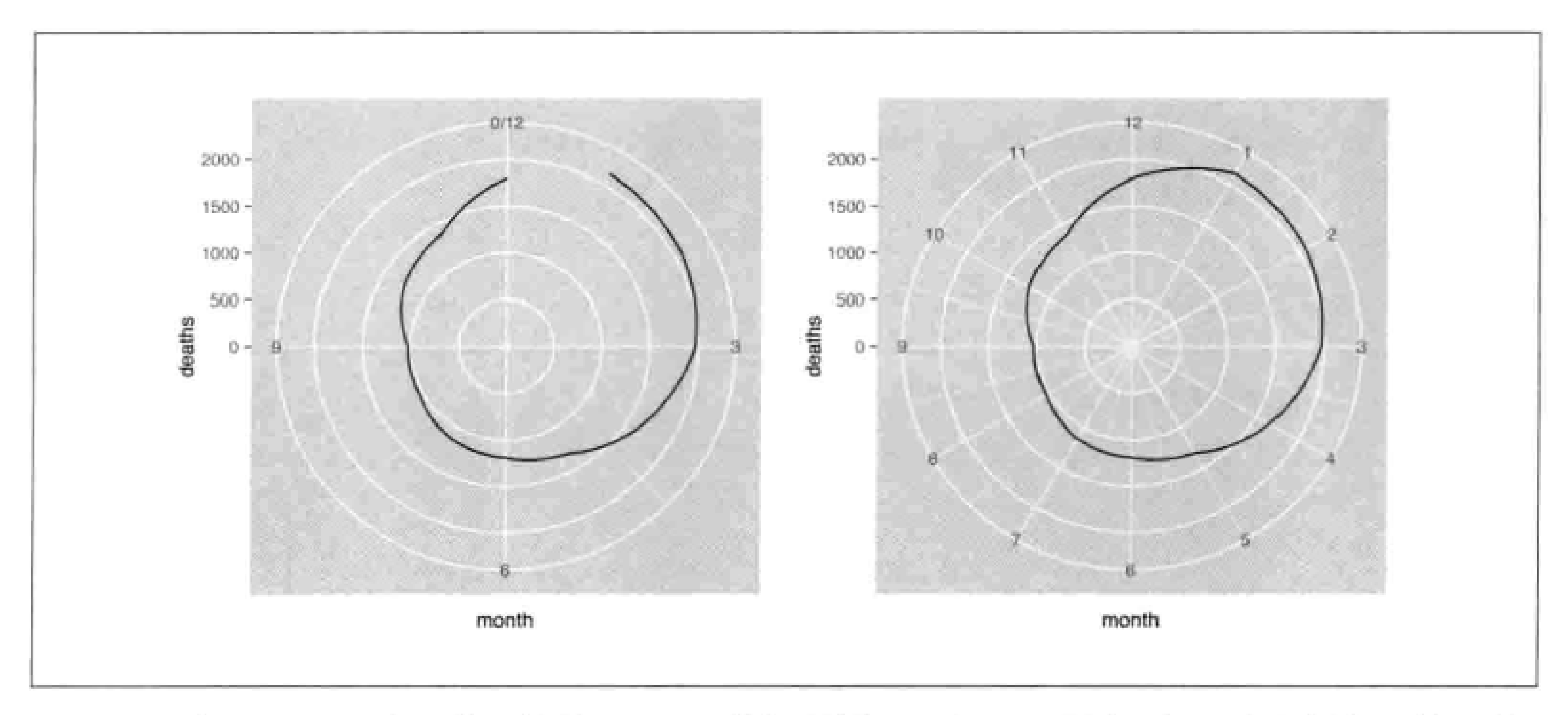


图 8-36　左图：theta 表示的 x 值从 0 到 12 的极坐标图　右图：通过添加一个虚拟的月份 0 数据点填补了缺口



注意运算符 %+% 的使用。当你使用 %+% 向一个 ggplot 对象添加一个数据框时，它会替换 ggplot 对象中的默认数据框。在本例中，它将 p 中默认的数据框从 md 改为了 mdnew。

另见

参见 10.4 节了解更多关于反转图例方位的信息。

参见 8.6 节以了解更多关于指定哪些值将拥有刻度线（分割点）和刻度标签的方法。

8.17 在坐标轴上使用日期

问题

如何在坐标轴上使用日期？

方法

将一列类为 Date 的变量映射到 *x* 轴或 *y* 轴即可。本例中我们将使用 economics 数据集：

```
# 观察数据结构
str(economics)

'data.frame':  478 obs. of  6 variables:
 $ date    : Date, format: "1967-06-30" "1967-07-31" ...
 $ pce     : num  508 511 517 513 518 ...
 $ pop     : int  198712 198911 199113 199311 199498 199657 199808 199920 ...
 $ psavert : num  9.8 9.8 9 9.8 9.7 9.4 9 9.5 8.9 9.6 ...
 $ uempmed : num  4.5 4.7 4.6 4.9 4.7 4.8 5.1 4.5 4.1 4.6 ...
 $ unemploy: int  2944 2945 2958 3143 3066 3018 2878 3001 2877 2709 ...
```

date 列是一个类为 Date 的对象，将其映射到 x 所得的结果如图 8-37 所示：

```
ggplot(economics, aes(x=date, y=psavert)) + geom_line()
```

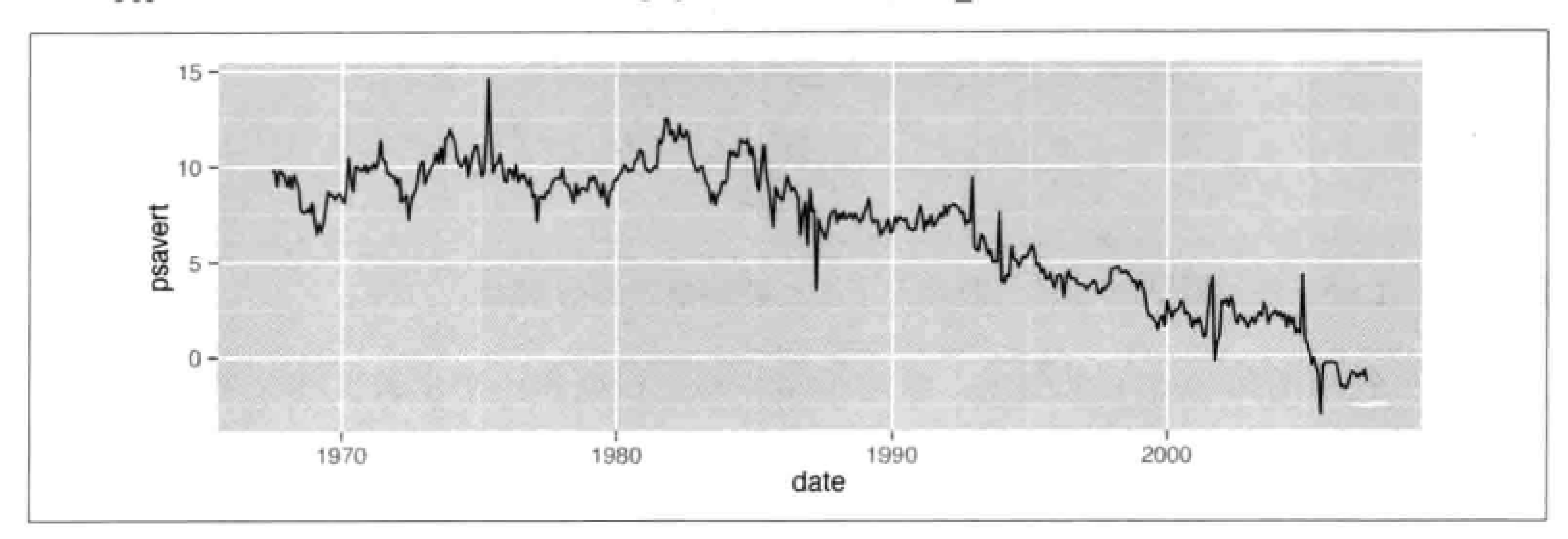


图 8-37 在 *x* 轴上显示日期

讨论

ggplot2 可以处理两类时间相关的对象：日期对象（类为 Date 的对象）和日期时间对象（类为 POSIXt 的对象）。两类对象的区别是，Date 对象表示的是日期，分辨率为一

天，而 POSIXt 对象则表示时刻，拥有精确到秒的小数部分的分辨率[①]。

设置分割点与数值坐标轴的方式类似——主要的不同在于设置所要使用的日期序列。这里我们将使用 economics 数据集从 1992 年年中到 1993 年年中的一个子集。如果未指定分割点，则将自动选择，如图 8-38 上图所示：

```
# 取 economics 的一个子集
econ <- subset(economics, date >= as.Date("1992-05-01") &
                          date <  as.Date("1993-06-01"))

# 基本图形 —— 不指定分割点
p <- ggplot(econ, aes(x=date, y=psavert)) + geom_line()
p
```

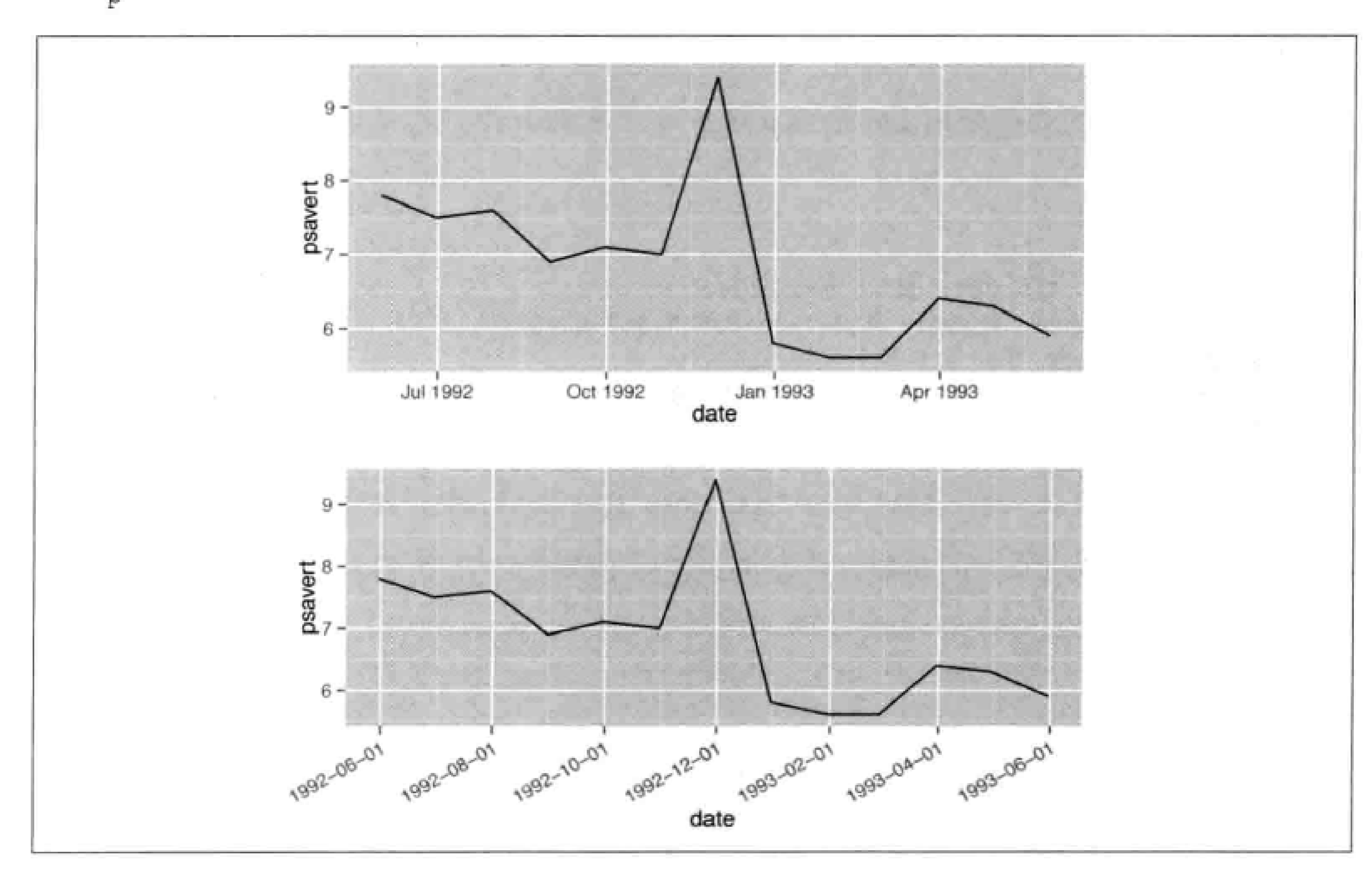


图 8-38　上图：使用默认的 *x* 轴分割点　下图：使用指定的分割点

分割点可使用函数 seq() 来创建，给定起始和终止日期和一个步长区间（见图 8-38 下图）即可：

```
# 指定一个日期向量为分割点
datebreaks <- seq(as.Date("1992-06-01"), as.Date("1993-06-01"), by="2 month")

# 使用分割点并旋转文本标签
p + scale_x_date(breaks=datebreaks) +
    theme(axis.text.x = element_text(angle=30, hjust=1))
```

① 在 R 中使用 %OSn 可以表示带有小数部分的秒数值，更多细节详见 ?strptime 中的说明。——译者注

注意，这里分割点（标签）的格式发生了改变，可以通过使用 scales 包中的 date_format() 函数来指定格式。这里我们将使用 "%Y %b"，结果的格式类似于“1992 Jun”，如图 8-39 所示[①]：

```
library(scales)
p + scale_x_date(breaks=datebreaks, labels=date_format("%Y %b")) +
    theme(axis.text.x = element_text(angle=30, hjust=1))
```

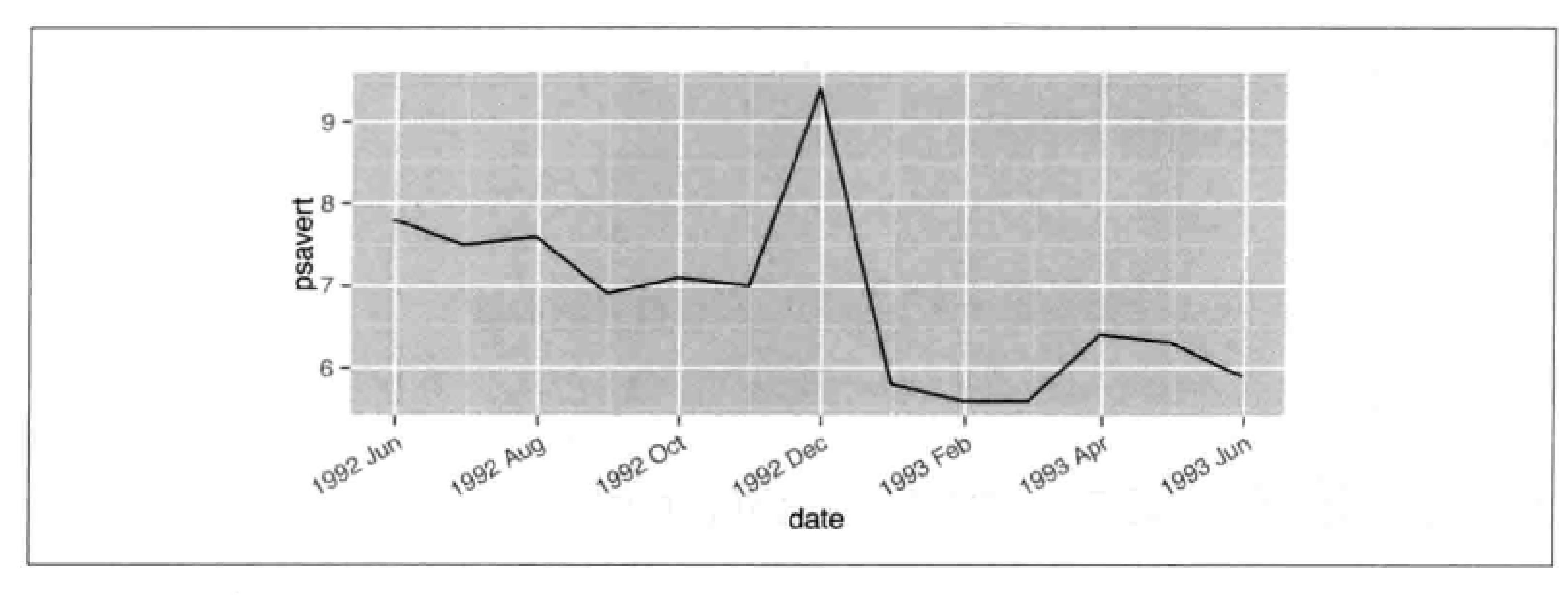


图 8-39　指定了日期格式的折线图

常用的日期格式选项列于表 8-1 中。它们应被放入一个字符串中传递给 date_format()，然后这些格式说明符就会被合适的值所替换。举例来说，如果你使用 "%B %d, %Y"，所得的标签将类似于“June 01, 1992”。

表 8-1　日期格式选项

选项	描述
%Y	含世纪的年份（2012）
%y	不含世纪的年份（12）[①]
%m	十进制数表示的月份（08）
%b	当前区域设置（locale）的月份名缩写（Aug）
%B	当前区域设置的月份名全称（August）
%d	十进制数表示的月份中的日期（04）
%U	十进制数表示的一年中的第几周，星期日作为每周的第一天（00–53）
%W	十进制数表示的一年中的第几周，星期一作为每周的第一天（00–53）
%w	星期几（0-6，星期日为 0）
%a	星期几的缩写名（Thu）
%A	星期几的全称（Thursday）

① 中文（系统）的日期时间格式下，输出结果的格式将为“1992　6 月”。要显示为此处的英文格式，使用 Sys.setlocale() 修改环境变量 LC_TIME 即可。详见 http://cos.name/cn/topic/109881。关于区域设置的更多说明，请参见下文。——译者注

以上选项中的一部分依赖于计算机的区域设置（locale）。月份和日期在不同的语言中会有不同的名称（本示例是使用美式区域设置生成的）。可以使用 Sys.setlocale() 来修改区域设置。例如，以下代码会将日期的格式修改为使用意大利的区域设置：

```
# Mac 和 Linux
Sys.setlocale("LC_TIME", "it_IT.UTF-8")

# Windows
Sys.setlocale("LC_TIME", "italian")
```

注意，区域的名称可能视平台的不同而有所区别，并且你的计算机必须支持这些在操作系统层面已经安装的区域设置。

另见

参见 ?Sys.setlocale 了解更多关于如何设定区域设置的信息。

参见 ?strptime 以了解关于如何将字符串转换为日期以及格式化日期输出的信息。

8.18 在坐标轴上使用相对时间

问题

如何在坐标轴上使用相对时间？

方法

时间值通常以数字的形式存储。举例来说，钟表上的时刻能够以一个表示小时的数字来存储。时间也能以从某个起始时间经过的分钟数或秒数来存储。在这些情况下，你应当将一个值映射到 x 轴或 y 轴上，并使用一个格式刷来生成合适的坐标轴标签（见图 8-40）：

```
# 转换时间序列对象 WWWusage 为数据框
www <- data.frame(minute = as.numeric(time(WWWusage)),
                  users = as.numeric(WWWusage))

# 定义一个格式刷函数——可将以分钟表示的时间转换为字符串
timeHM_formatter <- function(x) {
    h <- floor(x/60)
    m <- floor(x %% 60)
   lab <- sprintf("%d:%02d", h, m) # 将字符串格式化为 HH:MM（时：分）的格式
   return(lab)
}

# 默认的 x 轴
ggplot(www, aes(x=minute, y=users)) + geom_line()

# 使用格式化后的时间
ggplot(www, aes(x=minute, y=users)) + geom_line() +
    scale_x_continuous(name="time", breaks=seq(0, 100, by=10),
                       labels=timeHM_formatter)
```

① 00 ～ 68 的前缀将被补充为 20，69 ～ 99 的前缀将被补充为 19。详情参见 ?strptime 中对于 %y 的说明。——译者注

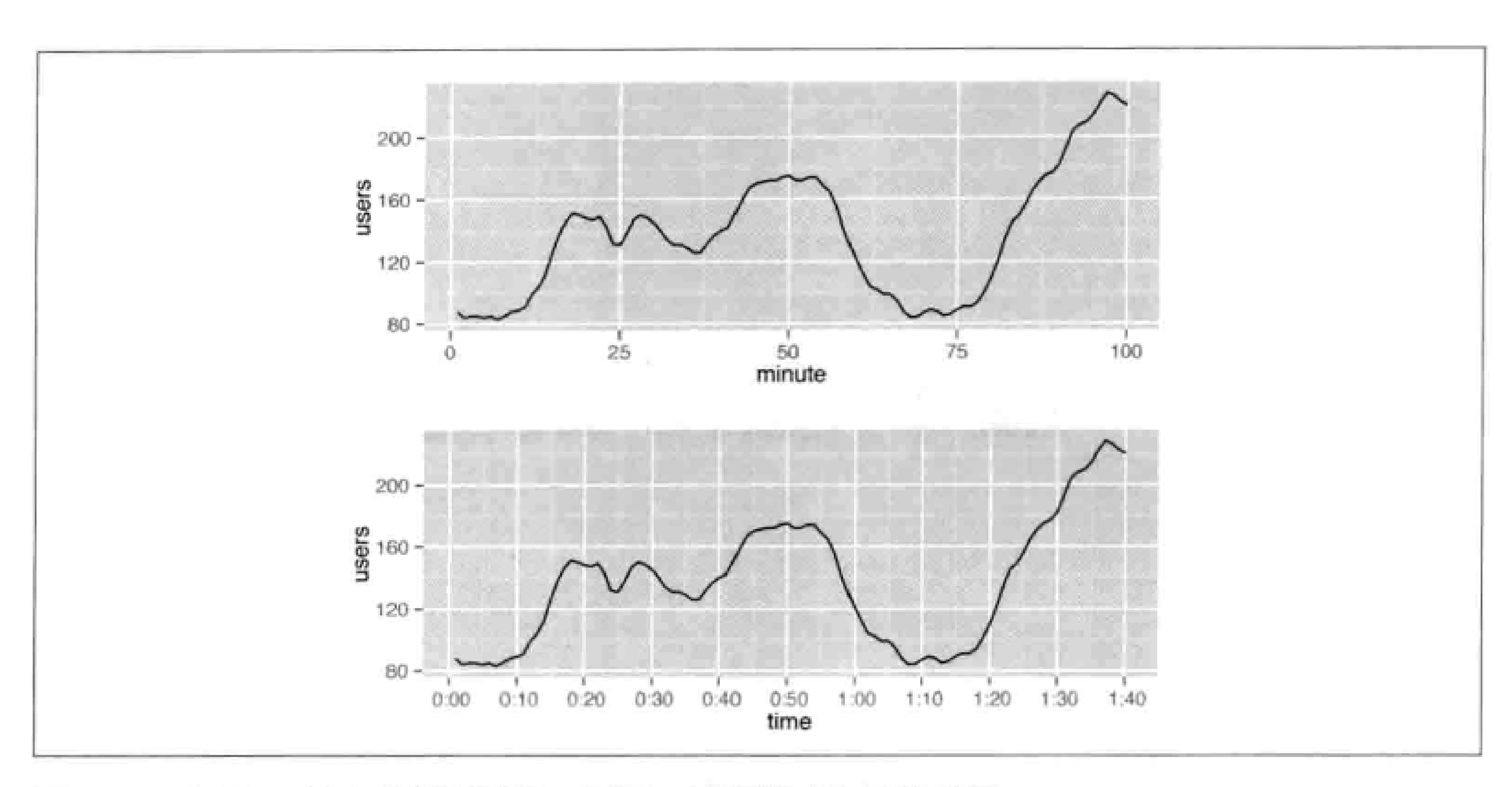


图 8-40　上图：x 轴上的相对时间　下图：使用格式化后的时间

讨论

在某些情况下，手动指定分割点和标签可能会简单一些，就像这样：

```
scale_x_continuous(breaks=c(0, 20, 40, 60, 80, 100),
    labels=c("0:00", "0:20", "0:40", "1:00", "1:20", "1:40"))
```

在前面的示例中，我们使用了函数 `timeHM_formatter()` 来将数值时间（以分钟表示）转换为一个类似于 `"1:10"` 的字符串：

```
timeHM_formatter(c(0, 50, 51, 59, 60, 130, 604))

"0:00"  "0:50"  "0:51"  "0:59"  "1:00"  "2:10"  "10:04"
```

要将其转换为 HH:MM:SS（时时 : 分分 : 秒秒）的格式，你可以使用以下格式刷函数：

```
timeHMS_formatter <- function(x) {
    h <- floor(x/3600)
    m <- floor((x/60) %% 60)
    s <- round(x %% 60)                          # 舍入到最接近的秒数
    lab <- sprintf("%02d:%02d:%02d", h, m, s)    # 格式化字符串为 HH:MM:SS 的格式
    lab <- sub("^00:", "", lab)                  # 如果开头存在 00: 则移除
    lab <- sub("^0", "", lab)                    # 如果开头存在 0 则移除
    return(lab)
}
```

使用一些示例数值运行它会得到：

```
timeHMS_formatter(c(20, 3000, 3075, 3559.2, 3600, 3606, 7813.8))

"0:20"   "50:00"   "51:15"   "59:19"   "1:00:00" "1:00:06" "2:10:14"
```

另见

参见 15.21 节以了解关于如何转换时间序列对象为数据框的信息。

第9章

控制图形的整体外观

在本章中，我们将讨论如何控制 ggplot2 图形的整体外观。作为 ggplot2 的基础，图形语法关注的是如何处理和展示数据——它并不关注如字体、背景色等问题。然而当我们呈现自己的数据时，很可能希望去调整这些元素的外观。ggplot2 的主题系统就为控制非数据元素的外观提供了可能。我们在前一章中提到了主题系统，在这一章中我们将更加详细地解释它的工作原理。

9.1 设置图形标题

问题

如何设置一幅图形的标题？

方法

使用 ggtitle() 设置标题，如图 9-1 所示：

```
library(gcookbook) # 为了使用数据集

p <- ggplot(heightweight, aes(x=ageYear, y=heightIn)) + geom_point()

p + ggtitle("Age and Height of Schoolchildren")

# 使用 \n 来换行
p + ggtitle("Age and Height\nof Schoolchildren")
```

讨论

使用 ggtitle() 与使用 labs(title = "标题文本") 是等价的。

如果你希望将标题移动到绘图区域内部，可以使用以下两种方法之一，这两种方法都

需要一点技巧（见图 9-2）。第一种方法是将一个负的 vjust 值与 ggtitle() 配合使用。这种方法的缺点是在绘图区域的上方仍然会留有空白的空间。

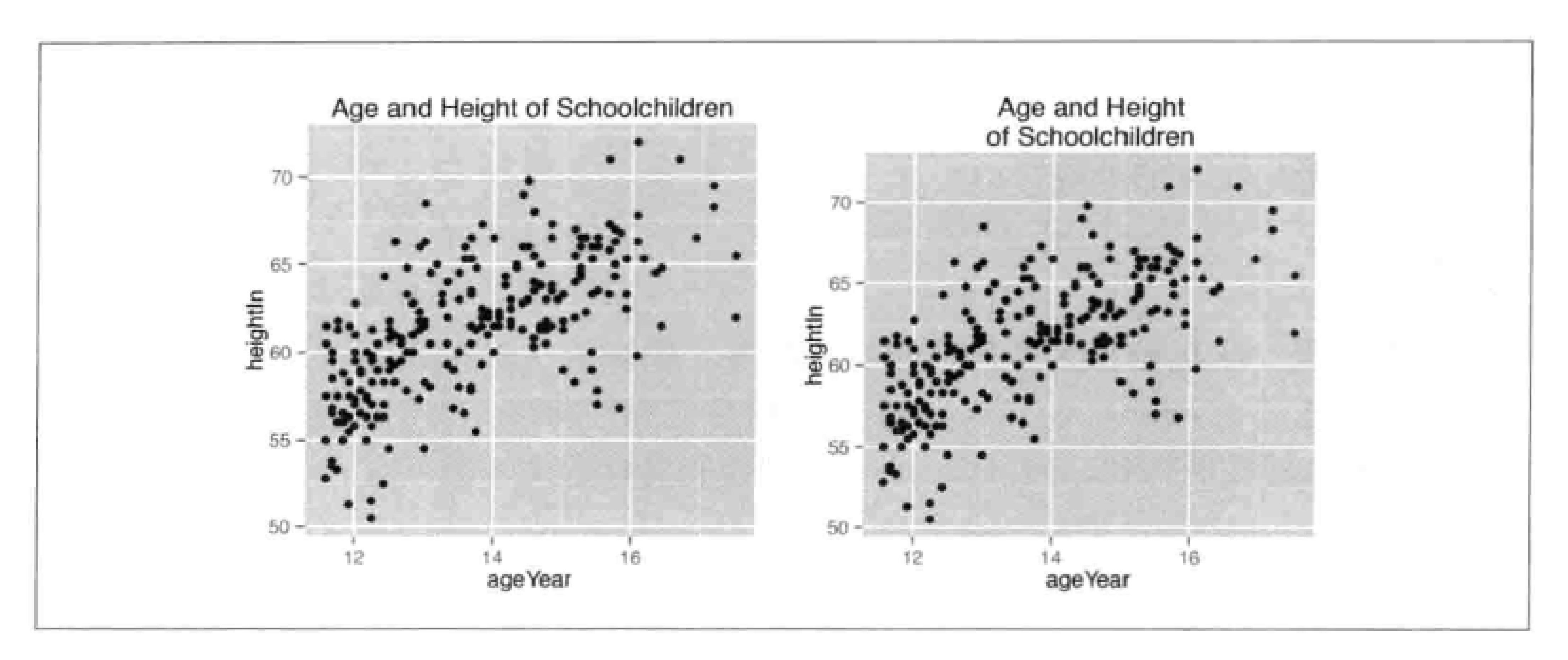


图 9-1　左图：添加了标题的散点图　右图：使用 \n 来换行

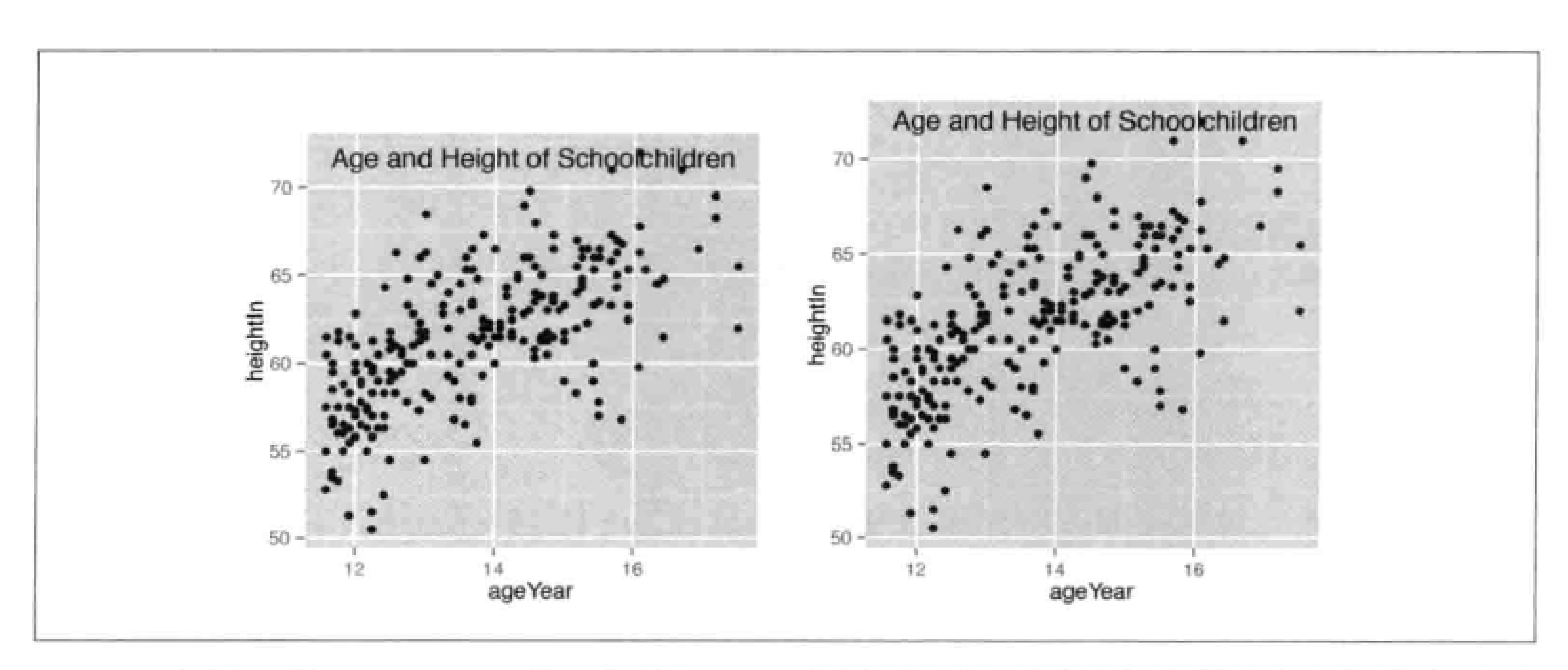


图 9-2　左图：使用 ggtitle() 和一个负的 vjust 值绘制的标题（注意绘图区域上方的额外空间）右图：在图形的顶部使用一个文本注解

第二种方式则是使用一个文本注解，设定其 *x* 的位置为 *x* 值域的中间，*y* 的位置为 Inf，这样就会将其置于绘图区域的顶部。这种方法同时需要 vjust 为正值，以使文本完全落入绘图区域：

```
# 移动标题到内部
p + ggtitle("Age and Height of Schoolchildren") +
    theme(plot.title=element_text(vjust = -2.5))

# 或使用一个文本型注解
p + annotate("text", x=mean(range(heightweight$ageYear)), y=Inf,
             label="Age and Height of Schoolchildren", vjust=1.5, size=6)
```

9.2 修改文本外观

问题

如何修改图形中文本的外观？

方法

要设置如标题、坐标轴标签和坐标轴刻度线等主题项目（theme item）的外观，使用 theme() 并通过 element_text() 设定对应项目的属性即可。举例来说，axis.title.x 控制着 *x* 轴标签的外观，而 plot.title 则控制着标题文本的外观（见图 9-3 左图）：

```
library(gcookbook)  # 为了使用数据集

# 基本图形
p <- ggplot(heightweight, aes(x=ageYear, y=heightIn)) + geom_point()

# 主题项目外观的控制
p + theme(axis.title.x=element_text(size=16, lineheight=.9, family="Times",
                                    face="bold.italic", colour="red"))

p + ggtitle("Age and Height\nof Schoolchildren") +
    theme(plot.title=element_text(size=rel(1.5), lineheight=.9, family="Times",
                                  face="bold.italic", colour="red"))

# rel(1.5) 表示字体大小将为当前主题基准字体大小的 1.5 倍
# 对于主题元素来说，字体大小（size）的单位为磅（pt）
```

要设置文本几何对象（即在图形内部使用 geom_text() 或 annotate() 添加的文本）的外观，只需设置其文本属性即可。举例来说（见图 9-3 右图）：

```
p + annotate("text", x=15, y=53, label="Some text", size = 7, family="Times",
        fontface="bold.italic", colour="red")

p + geom_text(aes(label=weightLb), size=4, family="Times", colour="red")

# 对于文本几何对象，字体大小的单位为毫米（mm）
```

讨论

在 ggplot2 中，文本项目分为两类：主题元素和文本几何对象。主题元素包括图形中的所有非数据元素：如标题、图例和坐标轴。文本几何对象则属于图形本身的一部分。

控制两类文本项目属性的参数略有不同，如表 9-1 所示。

表 9-1 主题元素和文本几何对象的文本属性

主题元素	文本几何对象	说明
family	family	Helvetica（无衬线）、Times（衬线）、Courier（等宽）
face	fontface	plain（普通）、bold（粗体）、italic（斜体）、bold.italic（粗斜体）
colour	colour	文字颜色（颜色名称或 "#RRGGBB" 形式的十六进制颜色代码）
size	size	字体大小（主题元素的单位是磅，几何对象的单位是毫米）
hjust	hjust	横向对齐：0= 左对齐，0.5= 居中，1= 右对齐
vjust	vjust	纵向对齐：0= 底部对齐，0.5= 居中，1= 顶部对齐
angle	angle	旋转角度，单位为度
lineheight	lineheight	行间距倍数

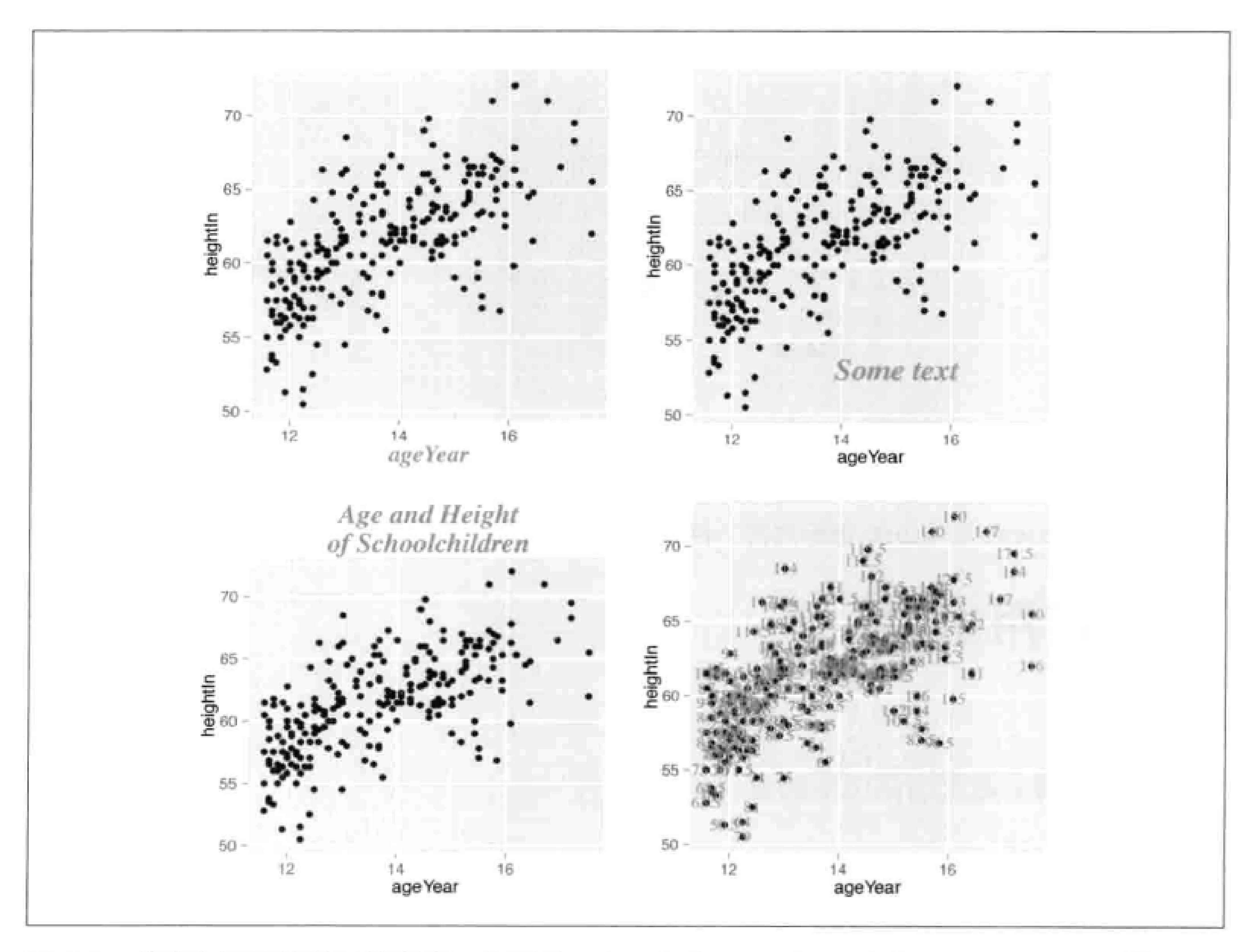


图 9-3 从左上图开始逆时针方向：分别对 axis.title.x、plot.title、geom_text() 和 annotate("text") 进行了设置

相应的主题元素已列于表 9-2 中，其中大多数理解起来都很简单直观。图 9-4 中展示了一部分。

表 9-2　theme() 中控制文本外观的主题项目

元素名称	说明
axis.title	双轴标签的外观
axis.title.x	x 轴标签的外观
axis.title.y	y 轴标签的外观
axis.ticks	双轴刻度标签的外观
axis.ticks.x	x 轴刻度标签的外观
axis.ticks.y	y 轴刻度标签的外观
legend.title	图例标题的外观
legend.text	图例项文本的外观
plot.title	图形总标题的外观
strip.text	双向分面标签的外观
strip.text.x	横向分面标签的外观
strip.text.y	纵向分面标签的外观

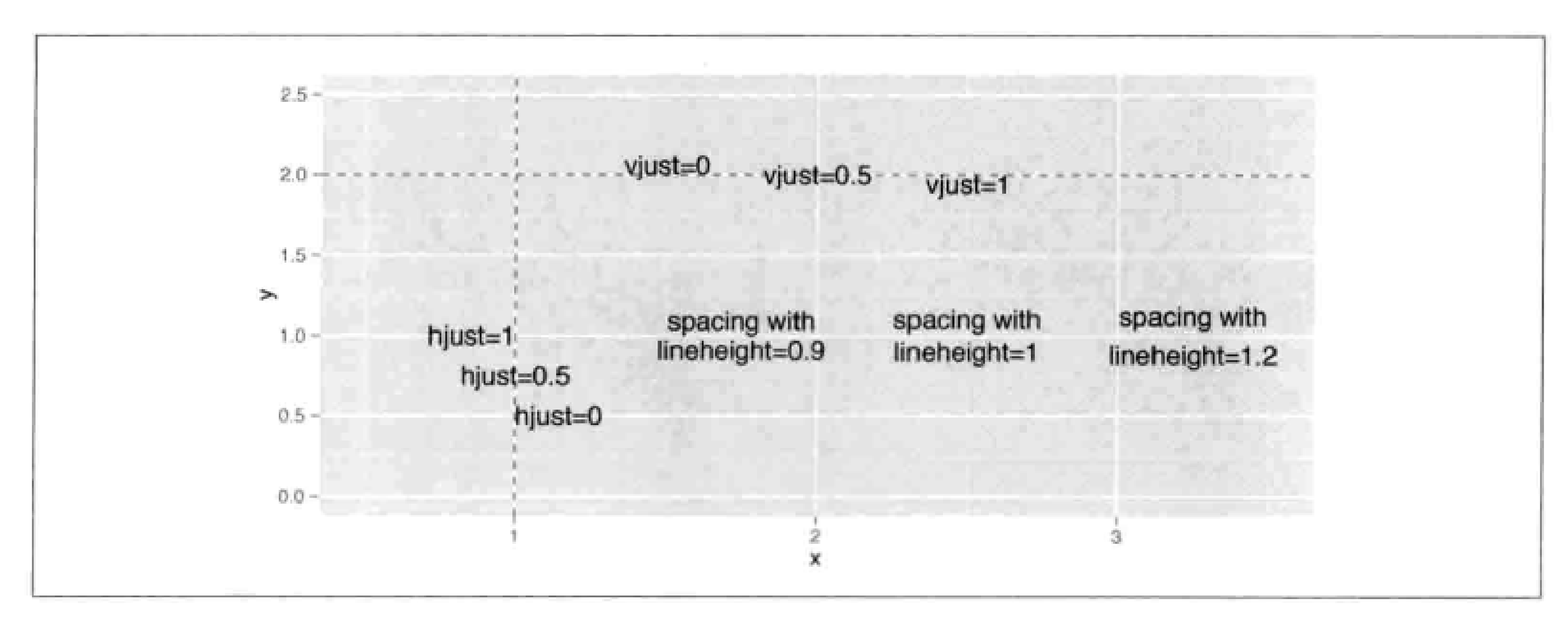


图 9-4　使用 hjust 和 vjust 进行对齐，并使用 lineheight 调整行间距

9.3　使用主题

问题

如何使用预制主题来控制图形的整体外观？

方法

要使用预制的主题，向图形添加 theme_bw() 或 theme_grey() 即可（见图 9-5）：

```
library(gcookbook)  # 为了使用数据集

# 基本图形
p <- ggplot(heightweight, aes(x=ageYear, y=heightIn)) + geom_point()

# (默认的) 灰色主题
```

```
p + theme_grey()

# 黑白主题
p + theme_bw()
```

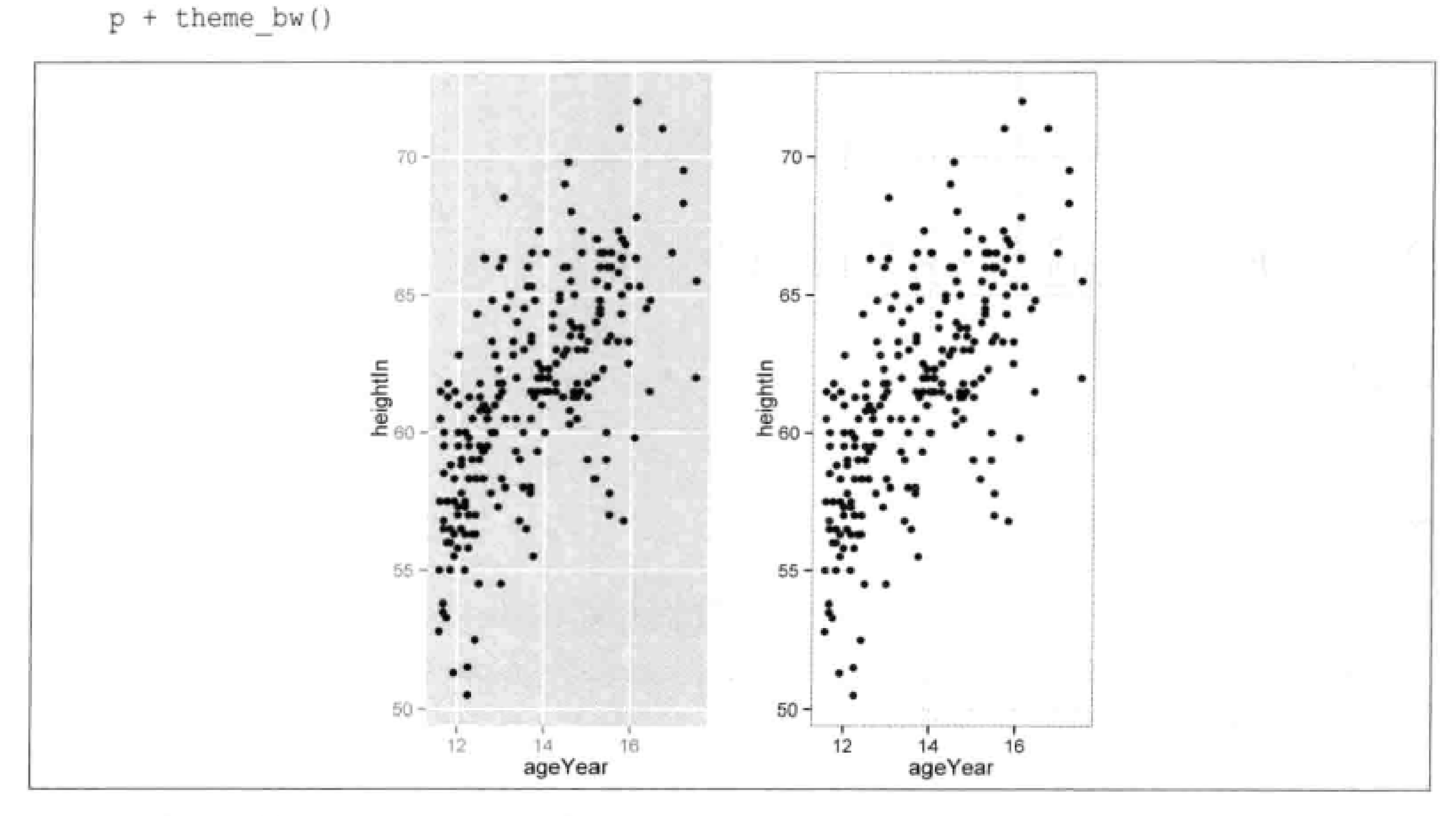


图 9-5　左图：使用 **theme_grey()**（默认主题）绘制的散点图　右图：使用 **theme_bw()** 绘制的散点图

讨论

ggplot2 中主题元素的某些常用属性是通过 `theme()` 来控制的。其中的多数属性，如标题、图例和坐标轴，位于绘图区域的外部，但另一些则位于绘图区域的内部，如网格线和背景色。

ggplot2 的两套自带主题是 `theme_grey()` 和 `theme_bw()`，不过你也可以创建自己的专属主题。

你可以自行设置两个内置主题的基本字体和字体大小（默认的基本字体为无衬线的 Helvetica，默认大小为 12）：

```
p + theme_grey(base_size=16, base_family="Times")
```

你可以使用 `theme_set()` 设置当前 R 会话下的默认主题：

```
# 为当前会话设置默认主题
theme_set(theme_bw())

# 将使用 theme_bw()
p

# 将默认主题重置回 theme_grey()
theme_set(theme_grey())
```

另见

要修改一套主题，参见 9.4 节。

要创建一套自定义主题，参见 9.5 节。

查看 ?theme 以了解所有可用的主题属性。

9.4 修改主题元素的外观

问题

如何修改主题元素的外观？

方法

要修改一套主题，配合相应的 element_xx 对象添加 theme() 函数即可。element_xx 对象包括 element_line、element_rect 以及 element_text。下列代码展示了许多常用主题属性的修改方法（见图 9-6）：

```
library(gcookbook)  # 为了使用数据集

# 基本图形
p <- ggplot(heightweight, aes(x=ageYear, y=heightIn, colour=sex)) + geom_point()

# 绘图区域的选项
p + theme(
    panel.grid.major = element_line(colour="red"),
    panel.grid.minor = element_line(colour="red", linetype="dashed", size=0.2),
    panel.background = element_rect(fill="lightblue"),
    panel.border = element_rect(colour="blue", fill=NA, size=2))

# 文本项目的选项
p + ggtitle("Plot title here") +
    theme(
    axis.title.x = element_text(colour="red", size=14),
    axis.text.x = element_text(colour="blue"),
    axis.title.y = element_text(colour="red", size=14, angle = 90),
    axis.text.y = element_text(colour="blue"),
    plot.title = element_text(colour="red", size=20, face="bold"))

# 图例选项
p + theme(
    legend.background = element_rect(fill="grey85", colour="red", size=1),
    legend.title = element_text(colour="blue", face="bold", size=14),
    legend.text = element_text(colour="red"),
    legend.key = element_rect(colour="blue", size=0.25))
```

```
# 分面选项
p + facet_grid(sex ~ .) + theme(
    strip.background = element_rect(fill="pink"),
    strip.text.y = element_text(size=14, angle=-90, face="bold"))
    # strip.text.x 同理，只不过是针对横向分面
```

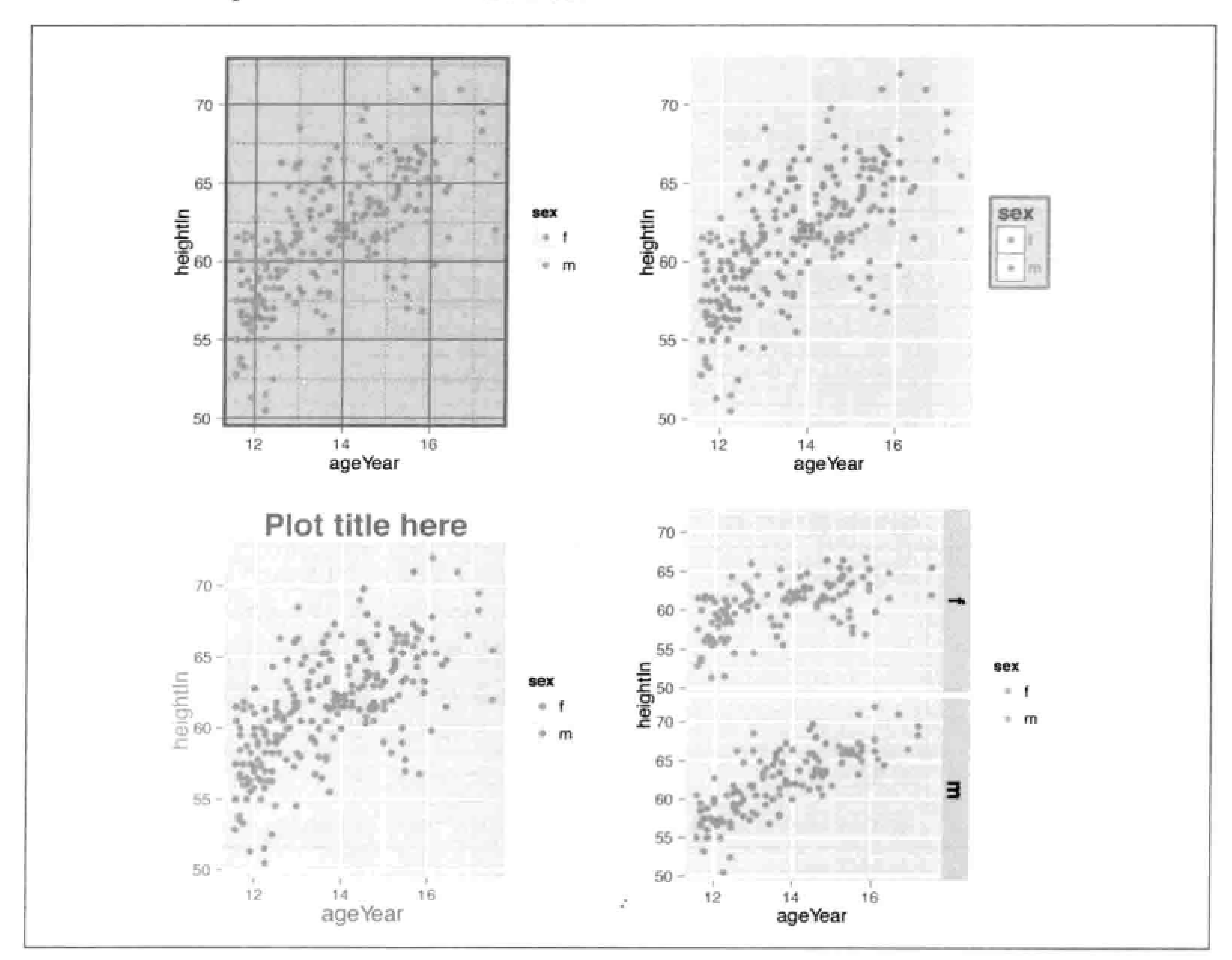


图 9-6 从左上图开始顺时针方向：修改绘图区域、图例、分面和文本项目的主题属性

讨论

如果你希望使用一套现成的主题并使用 `theme()` 微调其中的一些部分，则 `theme()` 必须接在指定主题的语句之后。否则，任何 `theme()` 的设定都将被你添加的主题所还原：

```
# 如果在添加一套完整的主题之前使用，theme() 将没有效果
p + theme(axis.title.x = element_text(colour="red")) + theme_bw()

# 在完整的主题后使用，theme() 可以正常工作
p + theme_bw() + theme(axis.title.x = element_text(colour="red", size=12))
```

许多常用的主题属性列于表 9-3 中。

表 9-3 在 theme() 中控制文本外观的主题项目

名称	说明	元素类型
text	所有文本元素	element_text()
rect	所有矩形元素	element_rect()
line	所有线条元素	element_line()
axis.line	坐标轴线	element_line()
axis.title	双轴标签的外观	element_text()
axis.title.x	*x* 轴标签的外观	element_text()
axis.title.y	*y* 轴标签的外观	element_text()
axis.text	双轴刻度标签的外观	element_text()
axis.text.x	*x* 轴刻度标签的外观	element_text()
axis.text.y	*y* 轴刻度标签的外观	element_text()
legend.background	图例的背景	element_rect()
legend.text	图例项文本的外观	element_text()
legend.title	图例标题的外观	element_text()
legend.position	图例的位置	"left"（左侧）、"right"（右侧）、"bottom"（下方）、"top"（上方），若希望放置在绘图区域内部，则需指定一个双元素数值向量（关于放置图例的更多知识，参见 10.2 节）
panel.background	绘图区域背景	element_rect()
panel.border	绘图区域周围的边框	element_rect(linetype="dashed")
panel.grid.major	主网格线	element_line()
panel.grid.major.x	纵向主网格线	element_line()
panel.grid.major.y	横向主网格线	element_line()
panel.grid.minor	次网格线	element_line()
panel.grid.minor.x	纵向次网格线	element_line()
panel.grid.minor.y	横向主网格线	element_line()
plot.background	整个图形的背景	element_rect(fill = "white", colour = NA)
plot.title	标题文本的外观	element_text()
strip.background	分面标签的背景	element_rect()
strip.text	纵向和横向分面标签的文本外观	element_text()
strip.text.x	横向分面标签的文本外观	element_text()
strip.text.y	纵向分面标签的文本外观	element_text()

9.5 创建自定义主题

问题

如何创建自定义主题?

方法

你可以通过向一套现成主题添加元素的方式创建自定义主题（见图 9-7）:

```
library(gcookbook)  # 为了使用数据集

# 从 theme_bw() 入手，修改一些细节
mytheme <- theme_bw() +
    theme(text       = element_text(colour="red"),
          axis.title = element_text(size = rel(1.25)))

# 基本图形
p <- ggplot(heightweight, aes(x=ageYear, y=heightIn)) + geom_point()

# 使用修改后的主题绘图
p + mytheme
```

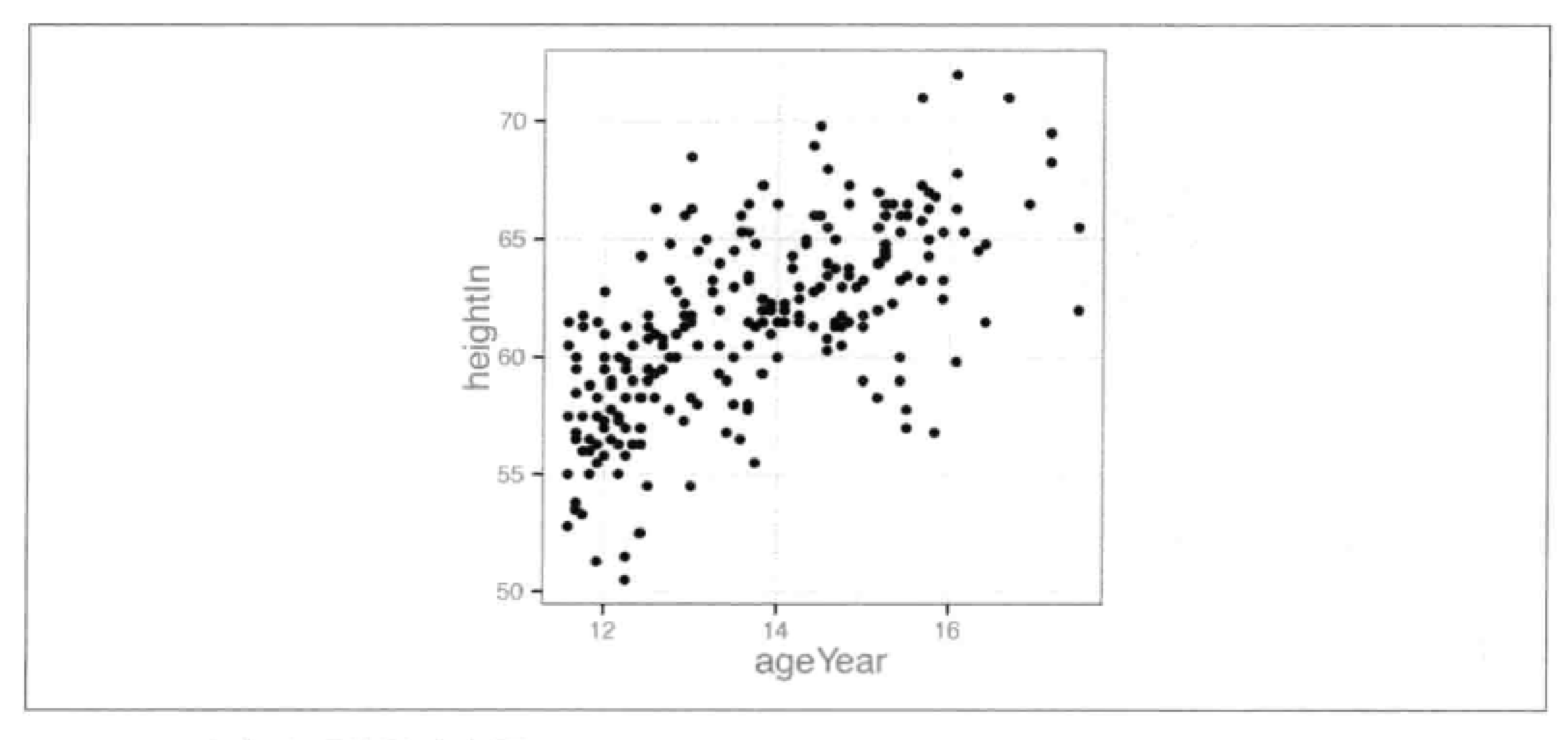


图 9-7 一套修改后的默认主题

讨论

使用 ggplot2，不仅可以直接利用默认主题，也可以修改这些主题以满足你的需求。可以添加新的主题元素或者修改现有的值，并将修改全部应用到多幅图形或者只应用到单幅图形上。

另见

主题的可修改选项列于 9.4 节中。

9.6 隐藏网格线

问题

如何隐藏图中的网格线？

方法

主网格线（与刻度线对齐的那些）可通过 panel.grid.major 来控制，次网格线（位于主网格线之间的那些）则通过 panel.grid.minor 来控制。以下代码将二者同时隐藏，如图 9-8 左图所示：

```
library(gcookbook)  # 为了使用数据集

p <- ggplot(heightweight, aes(x=ageYear, y=heightIn)) + geom_point()

p + theme(panel.grid.major = element_blank(),
          panel.grid.minor = element_blank())
```

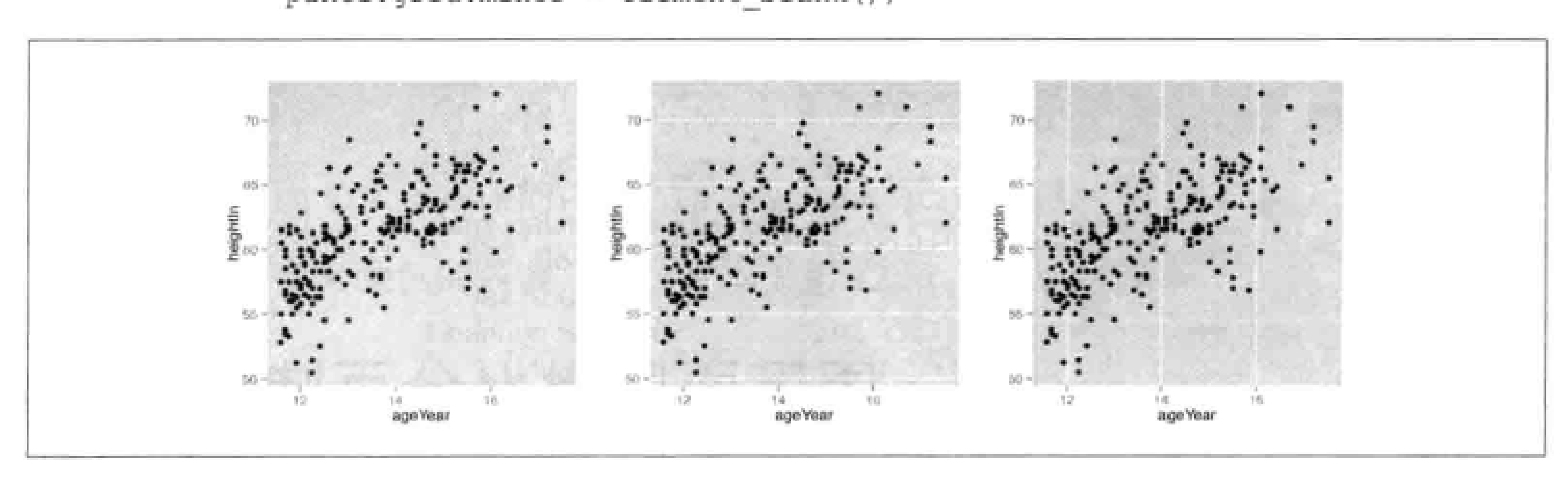


图 9-8　左图：无任何网格线　中图：无纵向网格线　右图：无横向网格线

讨论

通过使用 panel.grid.major.x、panel.grid.major.y、panel.grid.minor.x 和 panel.grid.minor.y，我们也可以只隐藏纵向或横向网格线，如图 9-8 中图和右图所示：

```
# 隐藏纵向网格线（与 x 轴交汇的那些）
p + theme(panel.grid.major.x = element_blank(),
          panel.grid.minor.x = element_blank())

# 隐藏横向网格线（与 y 轴交汇的那些）
p + theme(panel.grid.major.y = element_blank(),
          panel.grid.minor.y = element_blank())
```

第10章

图例

像 x 轴或 y 轴一样，图例也是一类引导元素：它可以向人们展示如何将视觉上的（图形）属性映射回数据本身。

10.1 移除图例

问题

如何从一幅图中移除图例？

方法

使用 guides()，并指定需要移除图例的标度（见图 10-1）：

```
# 基本图形（含图例）
p <- ggplot(PlantGrowth, aes(x=group, y=weight, fill=group)) + geom_boxplot()
```

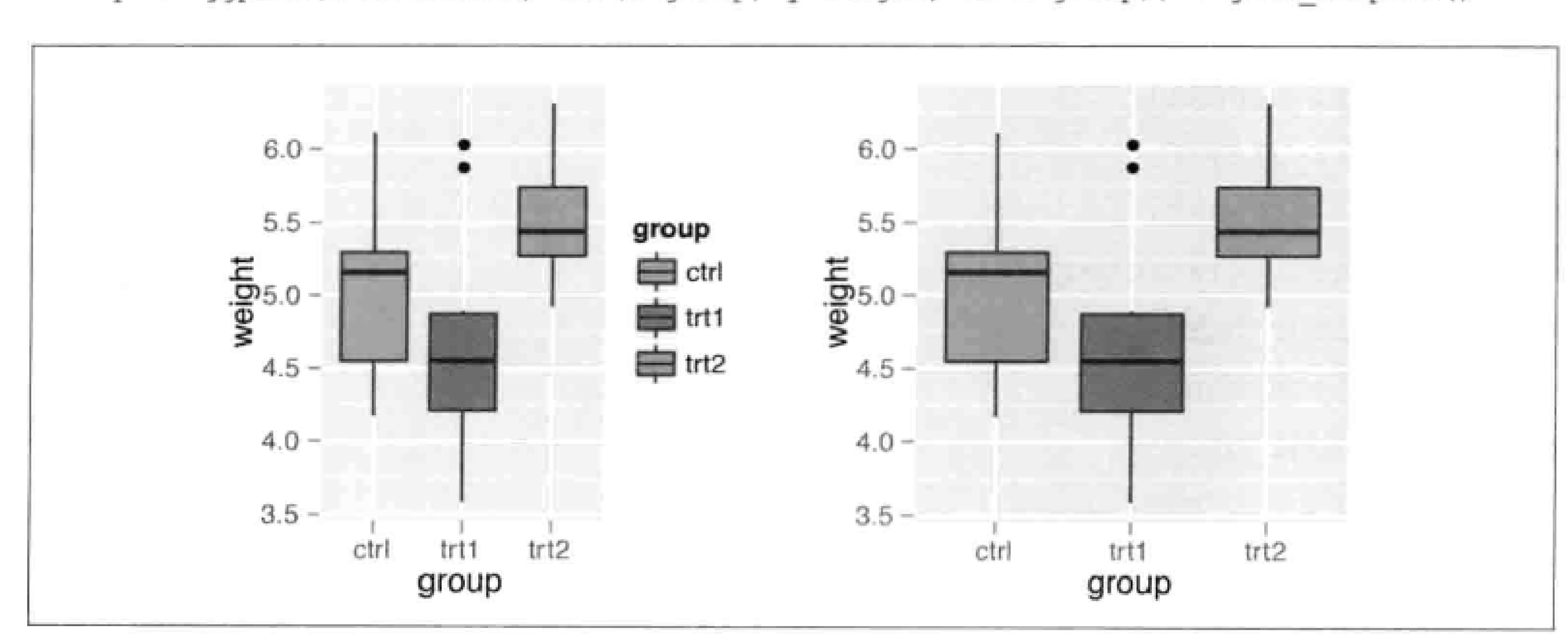


图 10-1 左图：默认外观 右图：移除图例后的外观

```
p

# 移除标度 fill 的图例
p + guides(fill=FALSE)
```

讨论

移除某个图例的另一种方式是在对应标度中设置 guide=FALSE。这样做得到的输出结果将与上述代码完全相同：

```
# 移除标度 fill 的图例
p + scale_fill_discrete(guide=FALSE)
```

还有一种移除图例的方法是使用主题系统。如果你有多于一种带有图例的图形属性映射（例如 color 和 shape），这样做将会移除所有图例：

```
p + theme(legend.position="none")
```

有些时候图例是冗余的，或者图例会在另一幅和当前图形共同展示的图形中提供。在这种情况下，从一幅图形中移除图例会比较有用。

在本例中，颜色提供了和 *x* 轴相同的信息，所以没有必要添加图例。还可以注意到，随着图例的移除，绘制数据的区域也随之变得更大。如果你希望得到与之前比例相同的绘图区域，则需要调整图形的整体尺寸。

当某个变量被映射到图形属性 fill 上时，默认使用的标度为 scale_fill_discrete()（与 scale_fill_hue() 等价），这会将不同的因子水平映射到色环上均匀分布的颜色值上。对于 fill 来说，也有其他的标度可用，如 scale_fill_manual()。如果你要使用其他图形属性的标度，如 colour（针对线和点）或 shape（针对点），则必须使用合适的对应标度。常用的标度包括：

- scale_fill_discrete()
- scale_fill_hue()
- scale_fill_manual()
- scale_fill_grey()
- scale_fill_brewer()
- scale_colour_discrete()
- scale_colour_hue()
- scale_colour_manual()
- scale_colour_grey()
- scale_colour_brewer()
- scale_shape_manual()
- scale_linetype()

10.2 修改图例的位置

问题

如何将默认处于右侧的图例移动到其他位置？

方法

使用 theme(legend.position=...) 即可。通过指定位置参数为 top、left、right 或 bottom，图例即可被放置在顶部、左侧、右侧或底部（见图 10-2 左图）：

```
p <- ggplot(PlantGrowth, aes(x=group, y=weight, fill=group)) + geom_boxplot() +
     scale_fill_brewer(palette="Pastel2")

p + theme(legend.position="top")
```

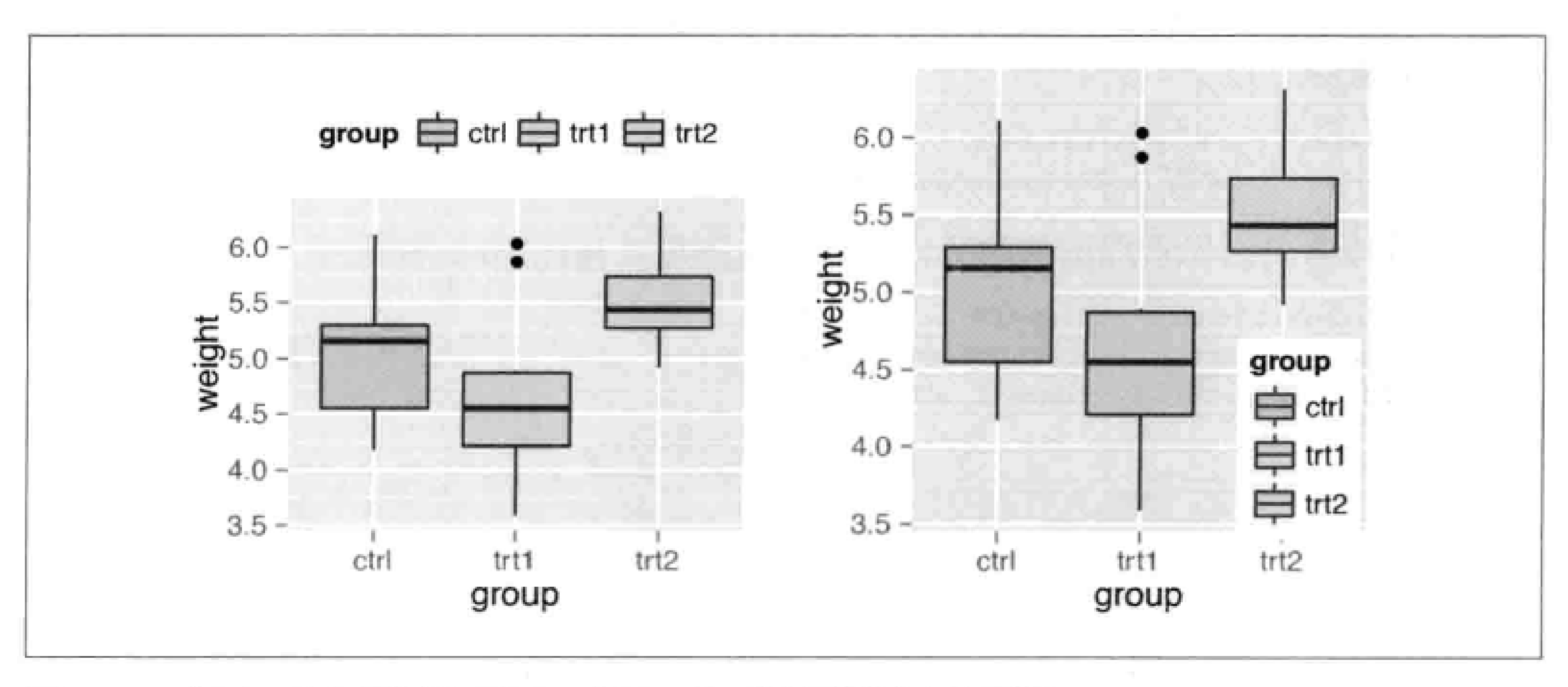


图 10-2　左图：位于顶部的图例　右图：位于绘图区域内的图例

通过指定像 legend.position=c(1,0) 这样的位置坐标，图例亦可被置于绘图区域内部（见图 10-2 右图）。坐标空间左下角为原点 (0, 0)，右上角为 (1, 1)。

讨论

你也可以使用 legend.justification 来指定图例框的哪一部分被放置到 legend.position 所指定的位置上。默认情况下，图例的中心 (0.5, 0.5) 被置于给定的坐标处，但是指定一个不同的点往往是有用的。

举例来说，以下代码将图例的右下角 (1, 0) 置于绘图区域的右下角 (1, 0)：

```
p + theme(legend.position=c(1,0), legend.justification=c(1,0))
```

而以下代码则会将图例的右上角置于绘图区域的右上角，如图 10-3 右图所示：

```
p + theme(legend.position=c(1,1), legend.justification=c(1,1))
```

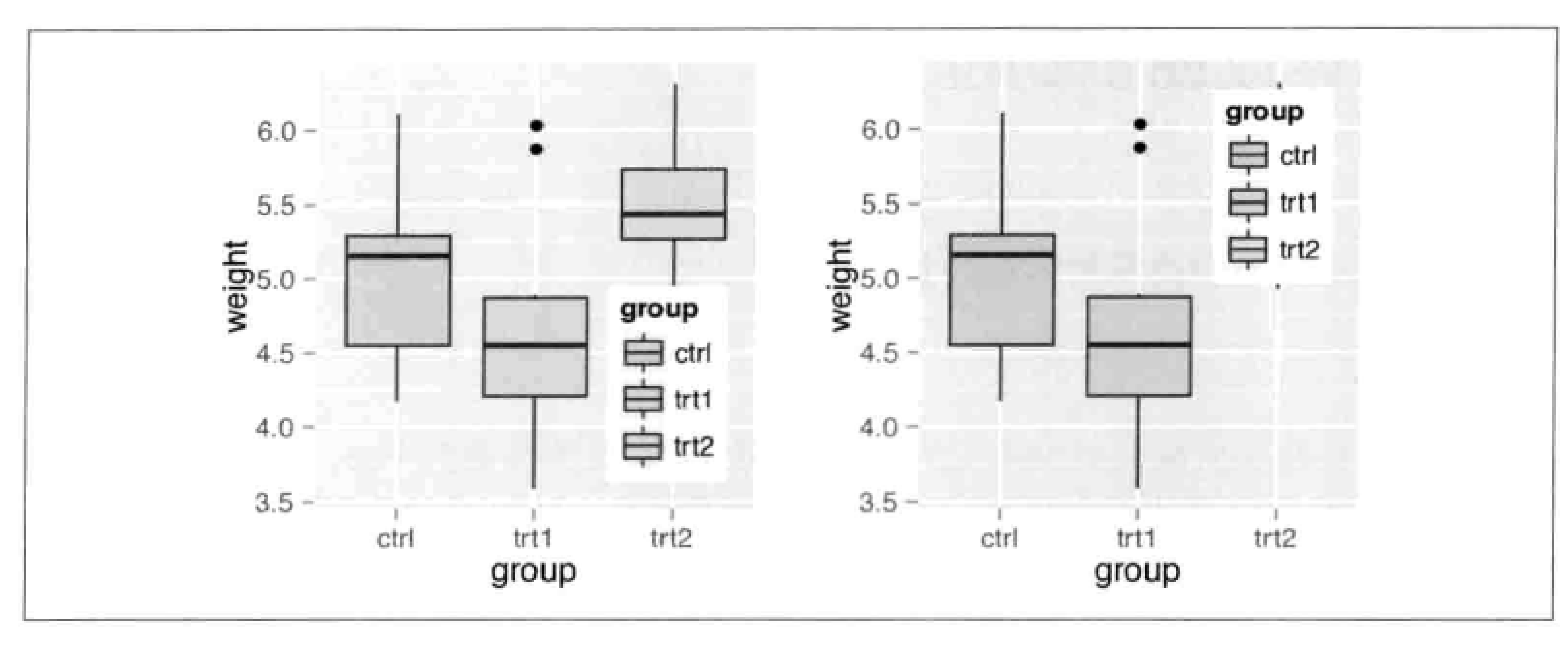


图 10-3　左图：位于右下角的图例　右图：位于右上角的图例

在绘图区域内放置图例时，添加一个不透明的边界使其与图形分开可能会有所帮助（见图 10-4 左图）：

```
p + theme(legend.position=c(.85,.2)) +
    theme(legend.background=element_rect(fill="white", colour="black"))
```

你也可以移除图例元素周围的边界以使其融入图形（见图 10-4 右图）：

```
p + theme(legend.position=c(.85,.2)) +
    theme(legend.background=element_blank()) +        # 移除整体的边框
    theme(legend.key=element_blank())                 # 移除每个图例项目周围的边框
```

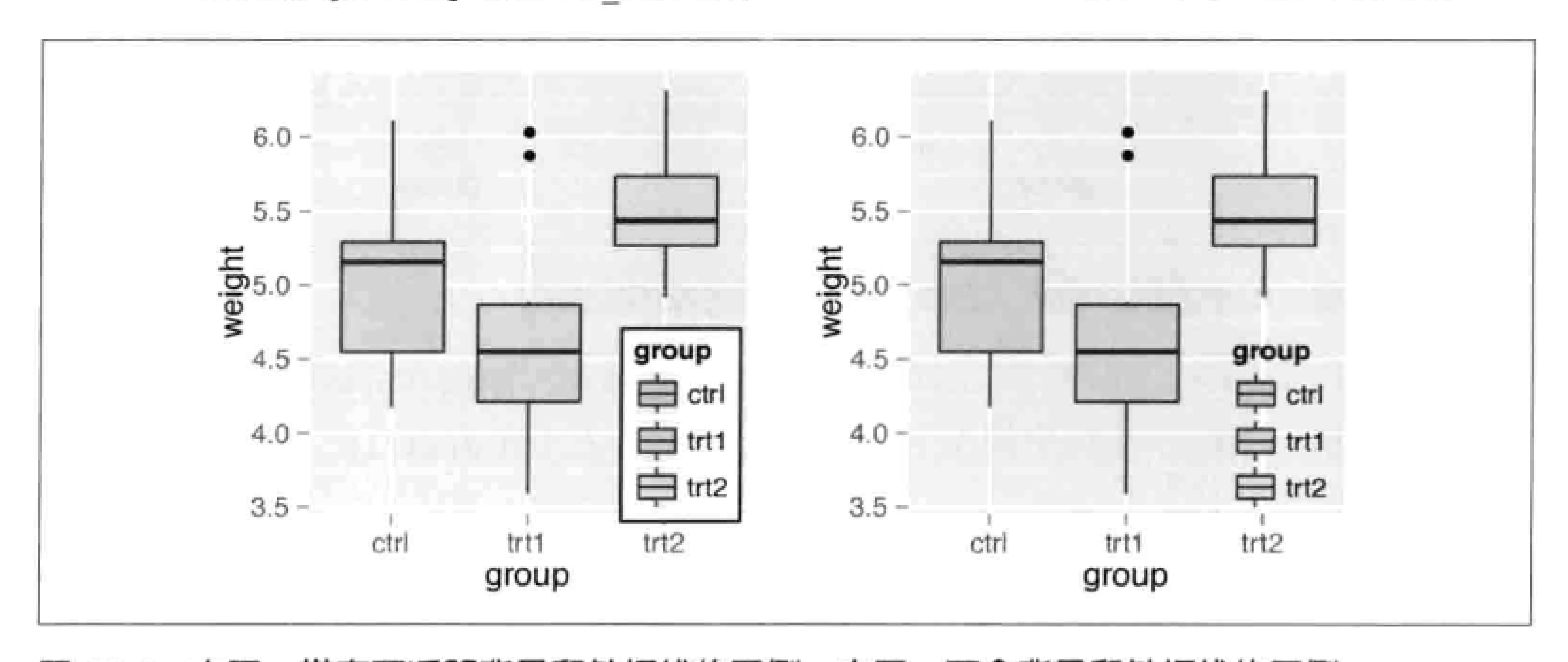


图 10-4　左图：带有不透明背景和外框线的图例　右图：不含背景和外框线的图例

10.3　修改图例项目的顺序

问题

如何修改图例中项目的顺序？

方法

将对应标度的参数 limits 设置为理想的顺序即可（见图 10-5）：

```
# 基本图形
p <- ggplot(PlantGrowth, aes(x=group, y=weight, fill=group)) + geom_boxplot()
p

# 修改项目顺序
p + scale_fill_discrete(limits=c("trt1", "trt2", "ctrl"))
```

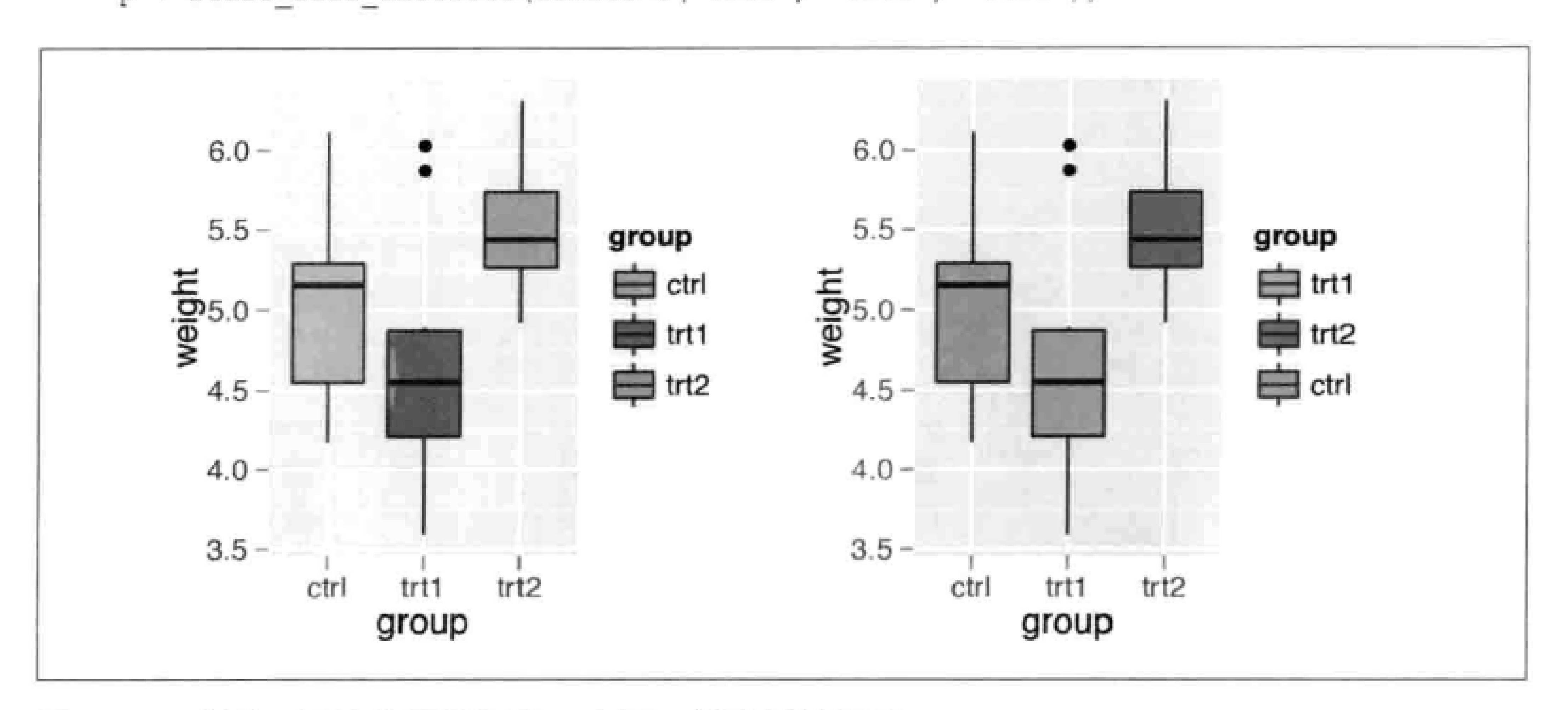


图 10-5　左图：默认的图例顺序　右图：修改后的顺序

讨论

要注意的是，*x* 轴上项目的顺序并没有改变。要修改这个顺序，需要设置 scale_x_discrete() 的 limits 参数（参见 8.4 节），或者修改数据，使其拥有一个不同的因子水平顺序（参见 15.8 节）。

在上例中，变量 group 被映射到了图形属性 fill 上。默认情况下，这将会使用 scale_fill_discrete()（与 scale_fill_hue() 的情况相同），将不同的因子水平映射到色环上均匀分布的颜色值上。然而，我们也可以使用其他的不同标度 scale_fill_xxx()。举例来说，我们可以使用灰度调色板（见图 10-6 左图）：

```
p + scale_fill_grey(start=.5, end=1, limits=c("trt1", "trt2", "ctrl"))
```

或者使用 RColorBrewer 中的调色板（见图 10-6 右图）：

```
p + scale_fill_brewer(palette="Pastel2", limits=c("trt1", "trt2", "ctrl"))
```

前述所有示例都是针对图形属性 fill 的。如果要使用其他图形属性的标度，如 colour（针对线和点）或 shape（针对点），则必须使用合适的对应标度。常用的标度包括：

- scale_fill_discrete()

- scale_fill_hue()
- scale_fill_manual()
- scale_fill_grey()
- scale_fill_brewer()
- scale_colour_discrete()
- scale_colour_hue()
- scale_colour_manual()
- scale_colour_grey()
- scale_colour_brewer()
- scale_shape_manual()
- scale_linetype()

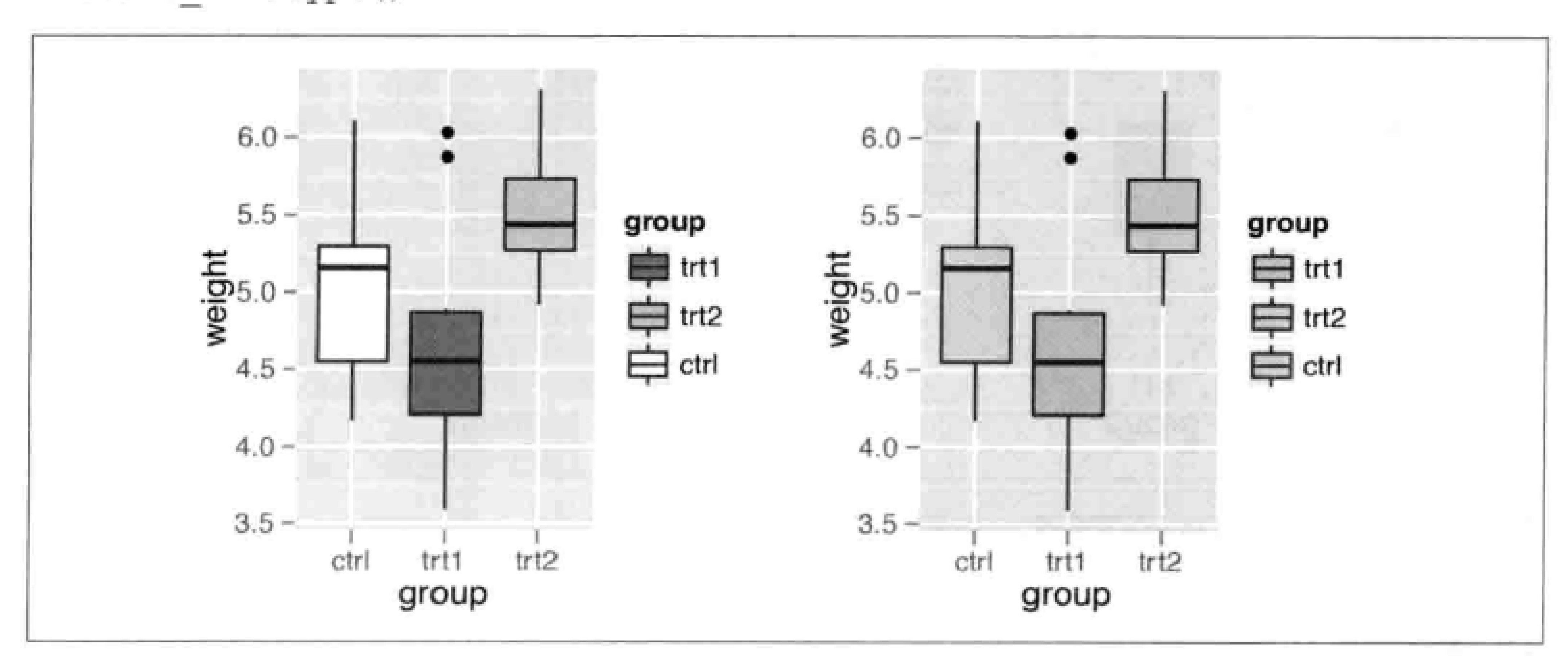


图 10-6　左图：使用灰度调色板并修改顺序　右图：使用 RColorBrewer 中的调色板

默认情况下，使用 scale_fill_discrete() 与使用 scale_fill_hue() 是等价的，这对颜色标度也成立。

另见

要反转图例顺序，参见 10.4 节。

要修改因子水平的顺序，参见 15.8 节。要根据其他变量的值对图例项目进行排序，参见 15.9 节。

10.4　反转图例项目的顺序

问题

如何反转图例中项目的顺序？

方法

添加 guides(fill=guide_legend(reverse=TRUE)) 以反转图例的顺序，如图 10-7 所示（对于其他的图形属性，使用相应图形属性的名称，如 colour 或 size 替换 fill 即可）：

```
# 基本图形
p <- ggplot(PlantGrowth, aes(x=group, y=weight, fill=group)) + geom_boxplot()
p

# 反转图例顺序
p + guides(fill=guide_legend(reverse=TRUE))
```

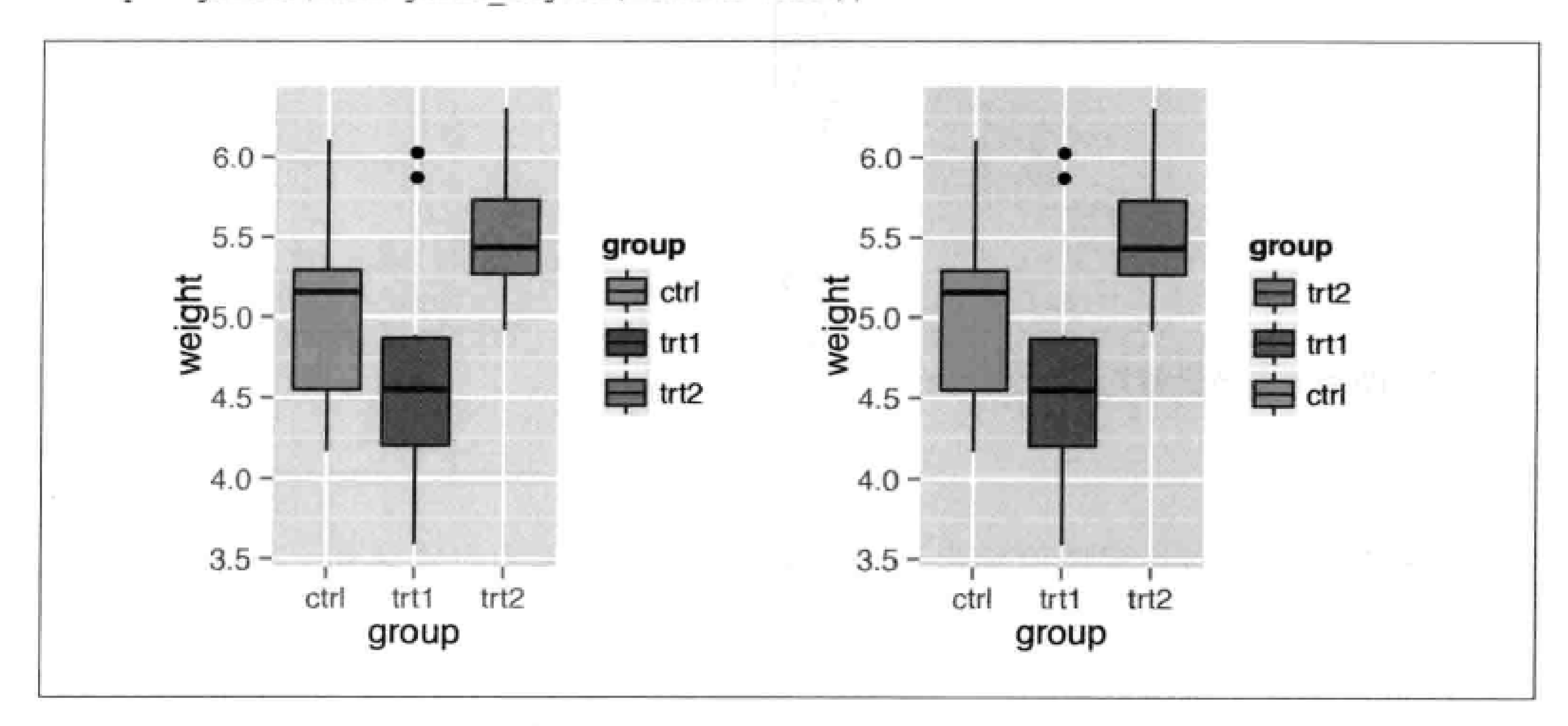


图 10-7　左图：默认的图例顺序　右图：反转后的顺序

讨论

在设定标度的同时也可以控制图例，如下所示：

```
scale_fill_hue(guide=guide_legend(reverse=TRUE))
```

10.5　修改图例标题

问题

如何修改图例标题中的文本？

方法

使用函数 labs() 并设定 fill、colour、shape 或任何对于图例来说合适的图形属性的值（见图 10-8）：

```
# 基本图形
p <- ggplot(PlantGrowth, aes(x=group, y=weight, fill=group)) + geom_boxplot()
```

```
p

# 设置图例标题为 "Condition"
p + labs(fill="Condition")
```



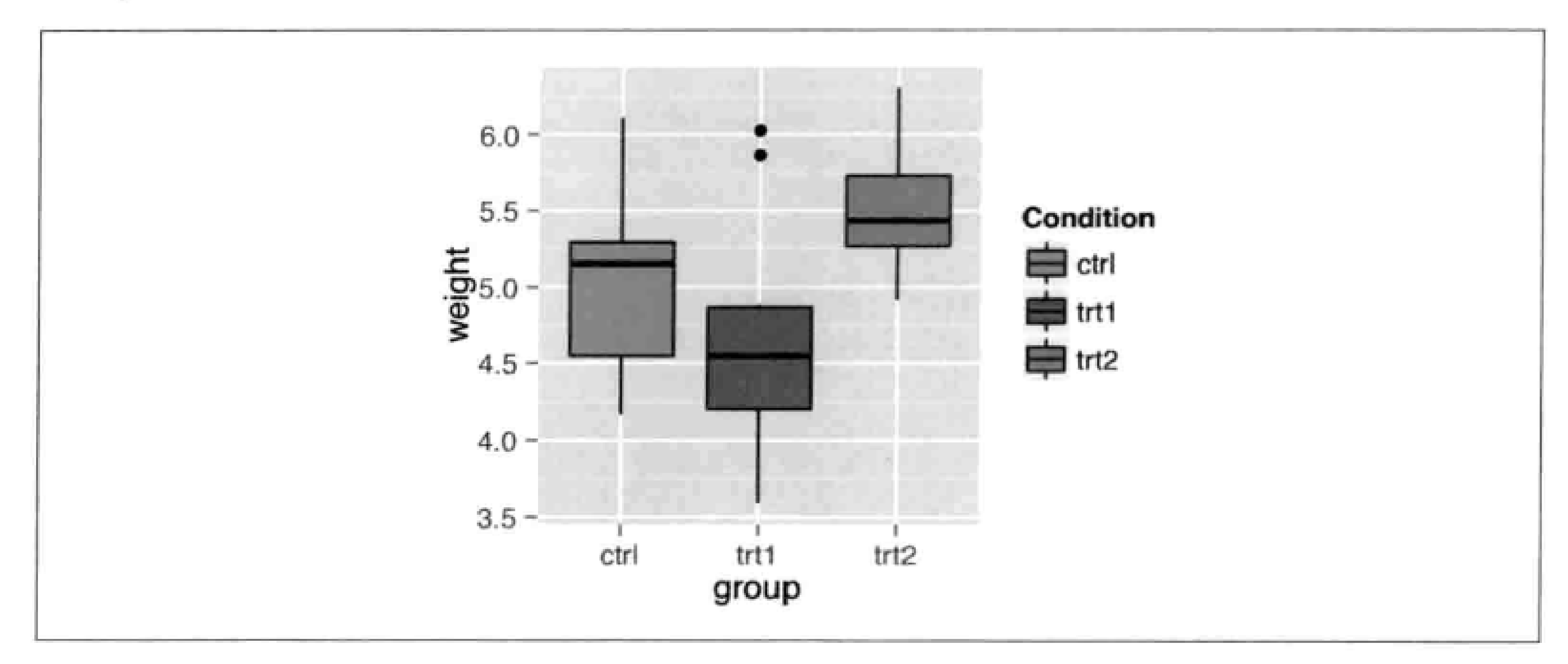


图 10-8　图例标题被设置为 "Condition"

讨论

在设定标度时也可以设置图例标题。由于图例和坐标轴均为引导元素，这样做与设置 *x* 轴或 *y* 轴标题的原理是相同的。

以下代码与上述代码效果相同：

```
p + scale_fill_discrete(name="Condition")
```

如果有多个变量被映射到带有图例的图形属性（即除 x 和 y 以外的图形属性），可以分别设置每个图例的标题。在本例中，我们将使用 \n 向其中一个标题添加一个换行（见图 10-9）：

```
library(gcookbook) # 为了使用数据集

# 绘制基本图形
hw <- ggplot(heightweight, aes(x=ageYear, y=heightIn, colour=sex)) +
      geom_point(aes(size=weightLb)) + scale_size_continuous(range=c(1,4))

hw

# 使用新的图例标题
hw + labs(colour="Male/Female", size="Weight\n(pounds)")
```

如果有一个变量被分别映射到两个图形属性，则默认会生成一个组合了两种情况的图例。例如，我们把 sex 同时映射到 shape 和 weight 上，将只会出现一个图例（见图 10-10 左图）：

```
hw1 <- ggplot(heightweight, aes(x=ageYear, y=heightIn, shape=sex, colour=sex)) +
       geom_point()

hw1
```

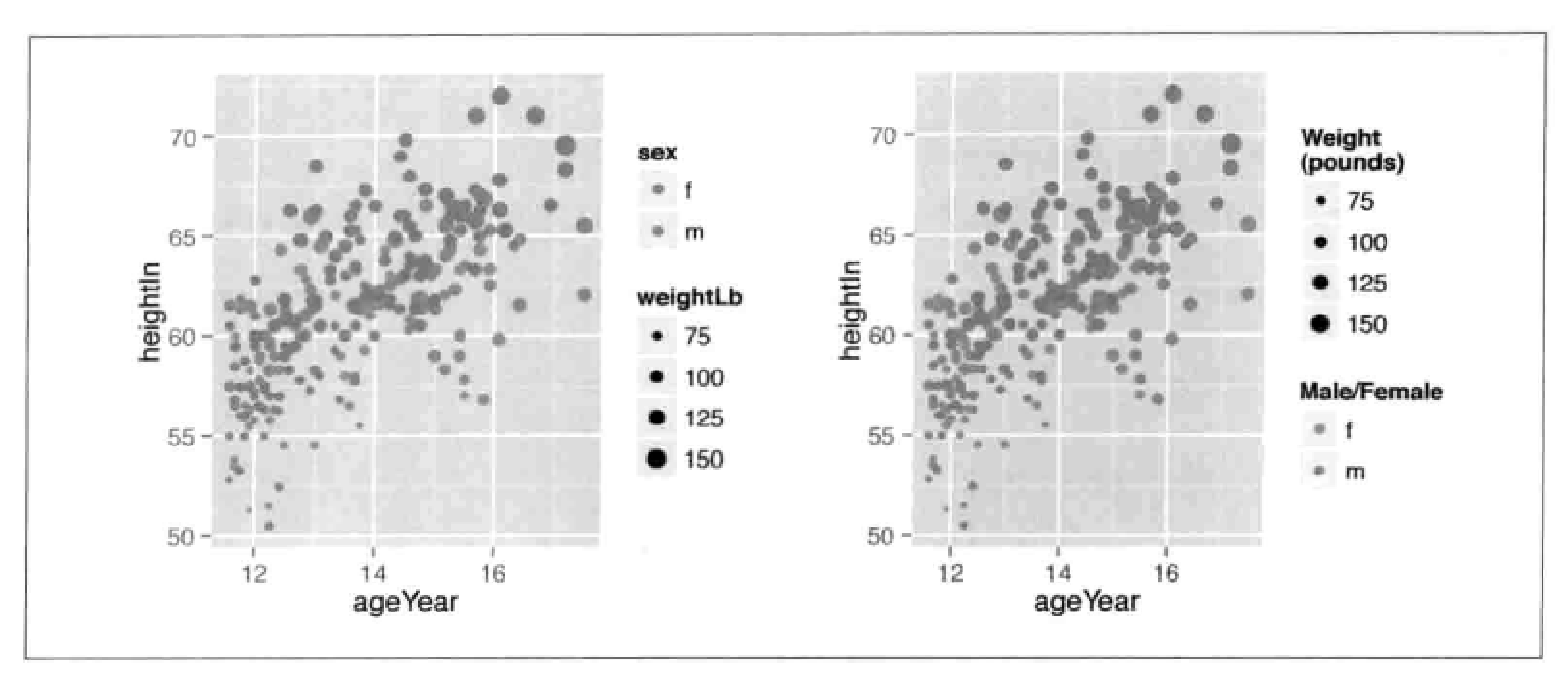


图 10-9　左图：使用原始标题的两个图例　右图：使用新标题的图例

要修改图例标题（见图 10-10 右图），你需要同时设置二者的标题。如果只修改其中一个，则会得到两个分离的图例（见图 10-10 中图）：

```
# 仅修改 shape 的标题
hwl + labs(shape="Male/Female")

# 同时修改 shape 和 colour 的标题
hwl + labs(shape="Male/Female", colour="Male/Female")
```

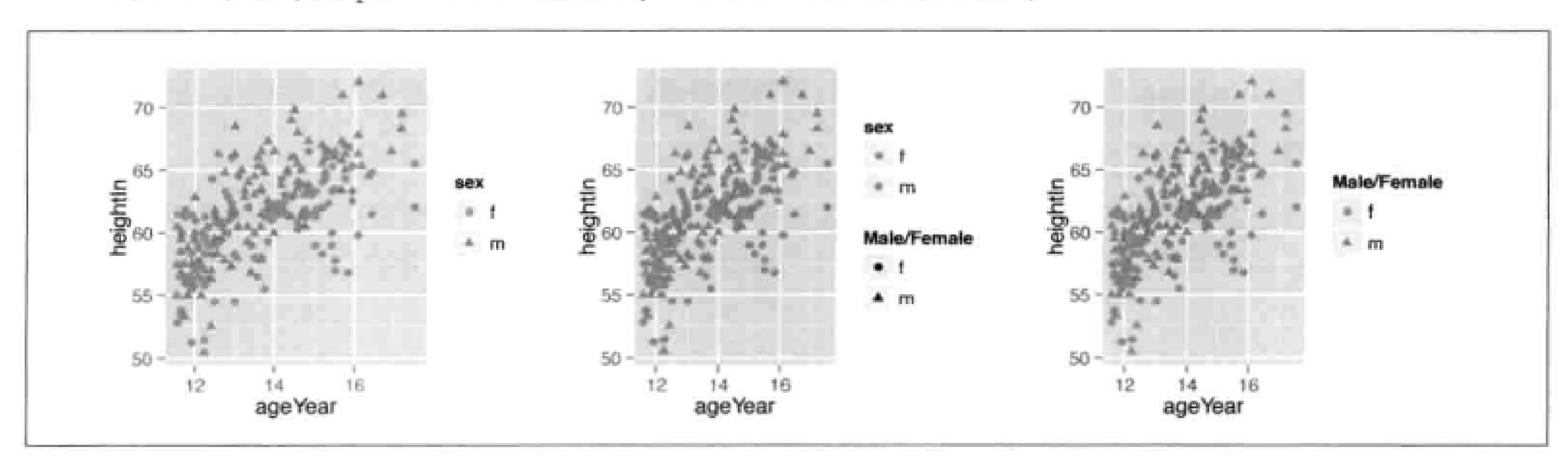


图 10-10　左图：一个变量映射到 shape 和 colour 时的默认图例　中图：重命名了 shape 对应的图例标题　右图：同时重命名了 shape 和 colour 对应的图例

我们也可以使用函数 guides() 来控制图例标题。这样做虽然有点啰嗦，不过在你使用它来控制其他属性的时候会比较有用：

```
p + guides(fill=guide_legend(title="Condition"))
```

10.6　修改图例标题的外观

问题

如何修改某个图例标题文本的外观？

方法

使用 theme(legend.title=element_text())（见图 10-11）：

```
p <- ggplot(PlantGrowth, aes(x=group, y=weight, fill=group)) + geom_boxplot()

p + theme(legend.title=element_text(face="italic", family="Times", colour="red",
                                     size=14))
```

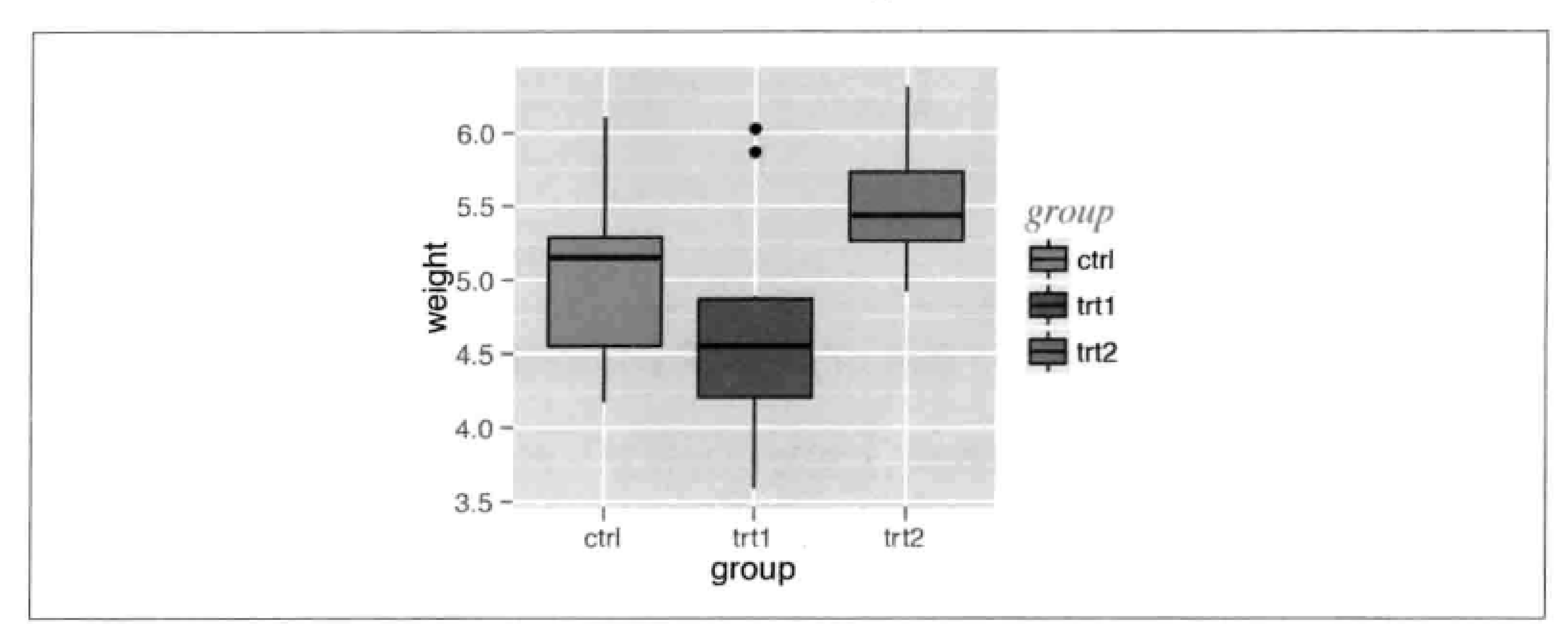


图 10-11　自定义图例标题外观

讨论

我们也可以通过 guides() 来指定图例标题的外观，但这种方式有点啰嗦。以下代码与上述代码效果相同：

```
p + guides(fill=guide_legend(title.theme=
           element_text(face="italic", family="times", colour="red", size=14)))
```

另见

参见 9.2 节以了解更多关于如何控制文本外观的信息。

10.7　移除图例标题

问题

如何移除某个图例的标题？

方法

添加语句 guides(fill=guide_legend(title=NULL)) 可以从图例中移除标题，如图 10-12 所示（对于其他图形属性，只需将 fill 替换为相应图形属性的名称，如 colour 或 size 即可）：

```
ggplot(PlantGrowth, aes(x=group, y=weight, fill=group)) + geom_boxplot() +
    guides(fill=guide_legend(title=NULL))
```

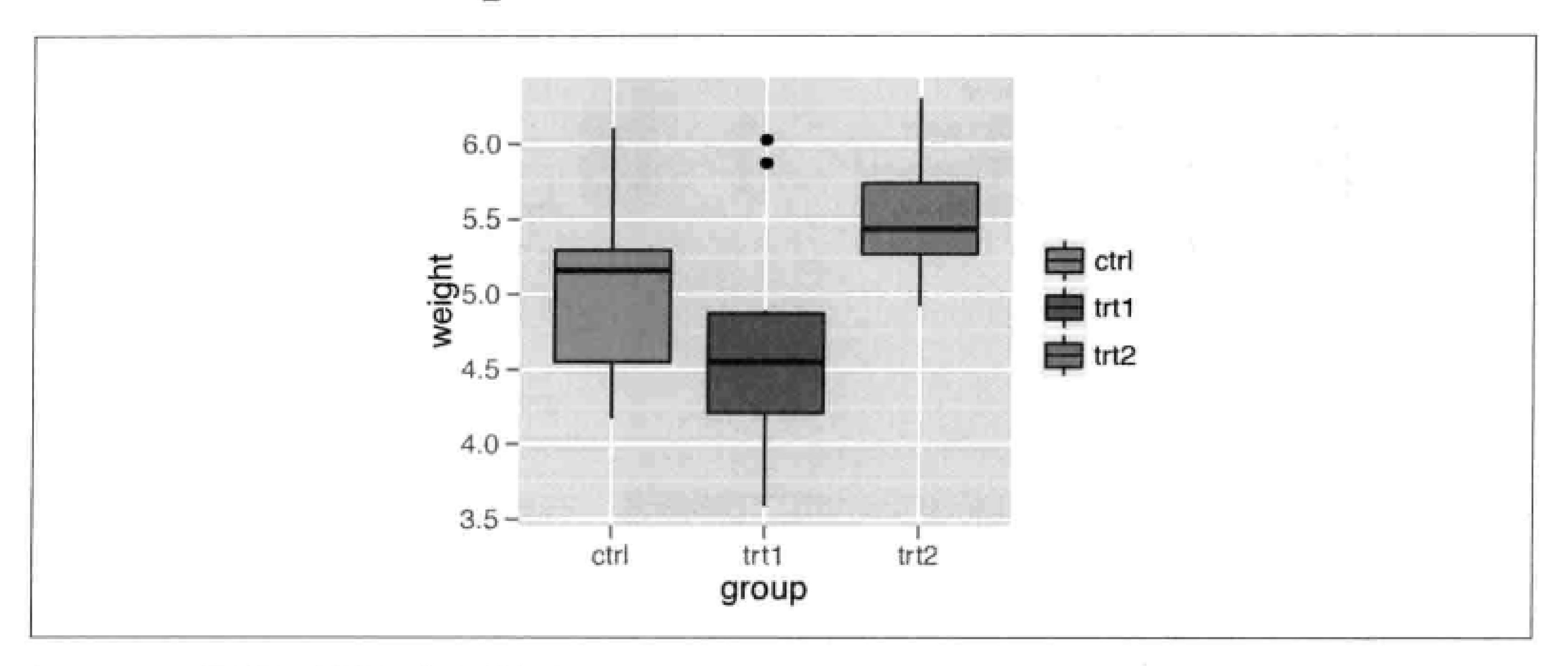


图 10-12　不含图例标题的箱线图

讨论

在设定标度的同时也可以控制图例标题。以下代码与前述代码效果相同：

```
scale_fill_hue(guide = guide_legend(title=NULL))
```

10.8　修改图例标签

问题

如何修改某个图例中的标签文本？

方法

设置标度中的 labels 参数即可（见图 10-13 左图）：

```
library(gcookbook) # 为了使用数据集

# 基本图形
p <- ggplot(PlantGrowth, aes(x=group, y=weight, fill=group)) + geom_boxplot()

# 修改图例标签
p + scale_fill_discrete(labels=c("Control","Treatment 1","Treatment 2"))
```

讨论

注意，*x* 轴的标签并没有改变。要修改它，需要设置 scale_x_discrete() 中的标签（参见 8.10 节），或者修改数据让其拥有不同的因子水平名称（参见 15.10 节）。

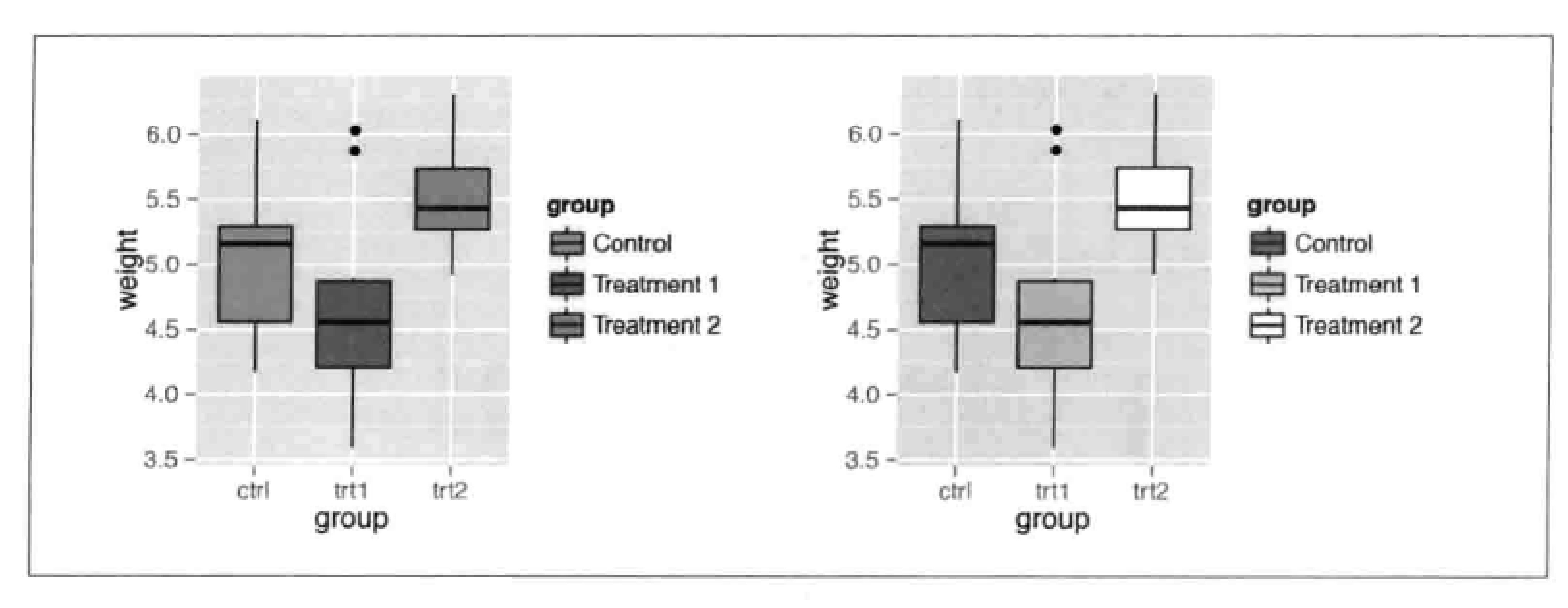


图 10-13　左图：通过默认的离散型标度手动指定了图例标签　右图：通过其他标度手动指定了标签

在上例中，变量 group 被映射到了图形属性 fill 上。默认情况下，这将会使用 scale_fill_discrete()，以将不同的因子水平映射到色环上均匀分布的颜色值上（与 scale_fill_hue() 的情况相同）。我们也可以使用其他 fill 标度通过相同的原理设定标签。举例来说，要绘制图 10-13 中的右图，只需：

```
p + scale_fill_grey(start=.5, end=1,
                    labels=c("Control","Treatment 1","Treatment 2"))
```

如果同时修改了图例项目的顺序，则标签会依照位置顺序与项目进行匹配。在本例中，我们将修改项目的顺序，并确保以相同的顺序设置标签（见图 10-14）：

```
p + scale_fill_discrete(limits=c("trt1", "trt2", "ctrl"),
                        labels=c("Treatment 1", "Treatment 2", "Control"))
```

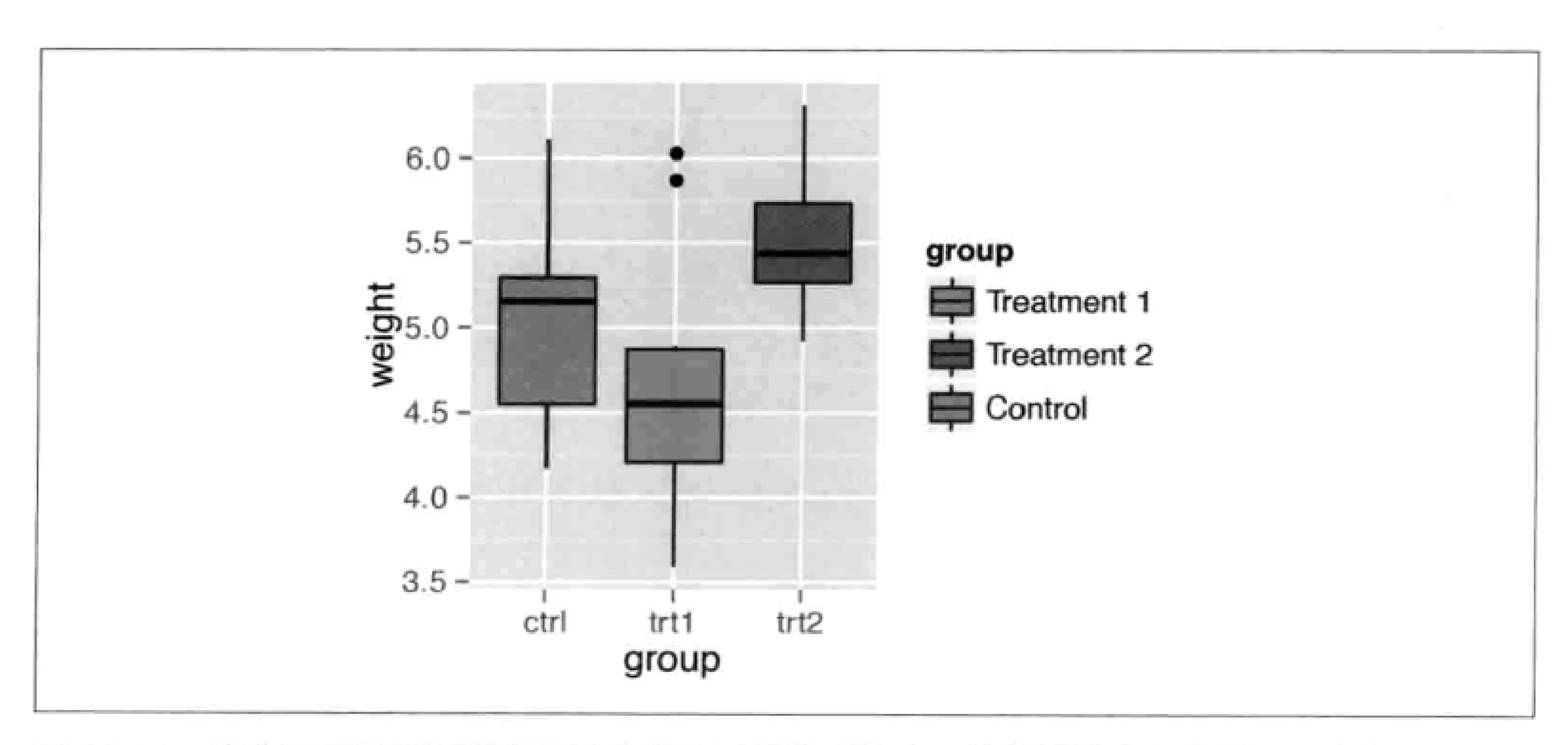


图 10-14　修改了图例标签顺序并手动指定了标签（注意 x 轴的标签和顺序并未改变）

如果有一个变量被分别映射到两个图形属性，则默认会生成一个组合了两种情况的图

例。如果希望修改图例标签，则必须同时修改两种标度中的标签；否则将得到两个分离的图例，如图 10-15 所示：

```
# 基本图形
p <- ggplot(heightweight, aes(x=ageYear, y=heightIn, shape=sex, colour=sex)) +
    geom_point()
p

# 修改一个标度中的标签
p + scale_shape_discrete(labels=c("Female", "Male"))

# 同时修改两个标度中的标签
p + scale_shape_discrete(labels=c("Female", "Male")) +
   scale_colour_discrete(labels=c("Female", "Male"))
```

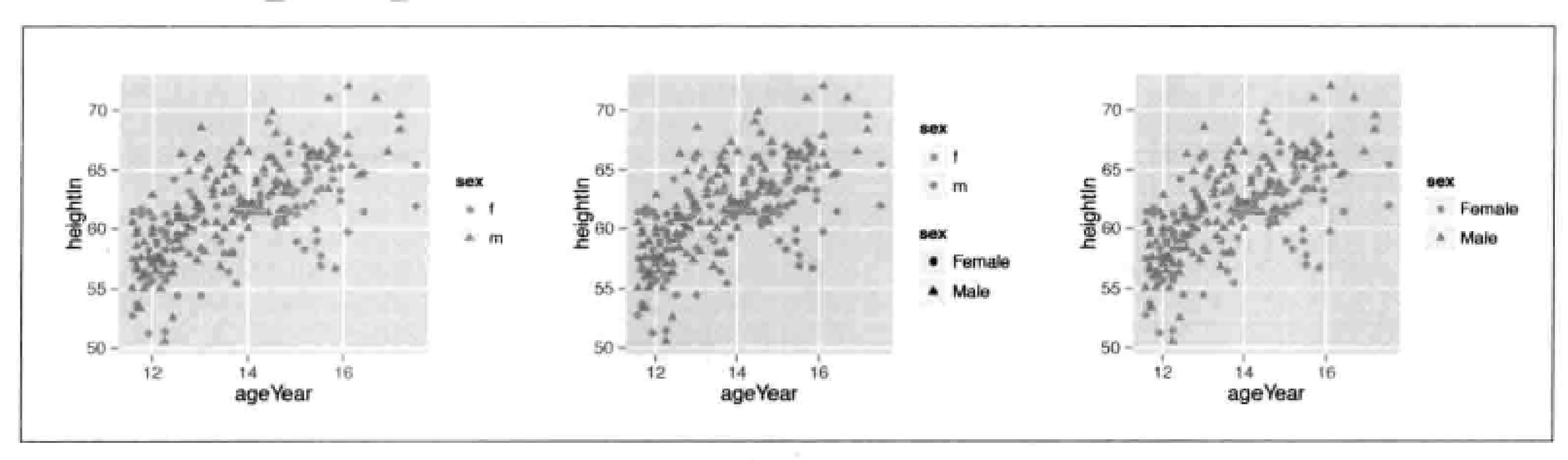


图 10-15　左图：将一个变量映射到 shape 和 colour　中图：为 shape 指定了新标签　右图：同时为 shape 和 colour 指定了新标签

其他含有图例的常用标度包括：

- `scale_fill_discrete()`
- `scale_fill_hue()`
- `scale_fill_manual()`
- `scale_fill_grey()`
- `scale_fill_brewer()`
- `scale_colour_discrete()`
- `scale_colour_hue()`
- `scale_colour_manual()`
- `scale_colour_grey()`
- `scale_colour_brewer()`
- `scale_shape_manual()`
- `scale_linetype()`

默认情况下，使用 `scale_fill_discrete()` 与使用 `scale_fill_hue()` 是等价的；这对颜色标度也成立。

10.9　修改图例标签的外观

问题

如何修改某个图例中标签的外观？

方法

使用 theme(legend.text=element_text())（见图 10-16）：

```
# 基本图形
p <- ggplot(PlantGrowth, aes(x=group, y=weight, fill=group)) + geom_boxplot()

# 修改图例标签的外观
p + theme(legend.text=element_text(face="italic", family="Times", colour="red",
                                   size=14))
```

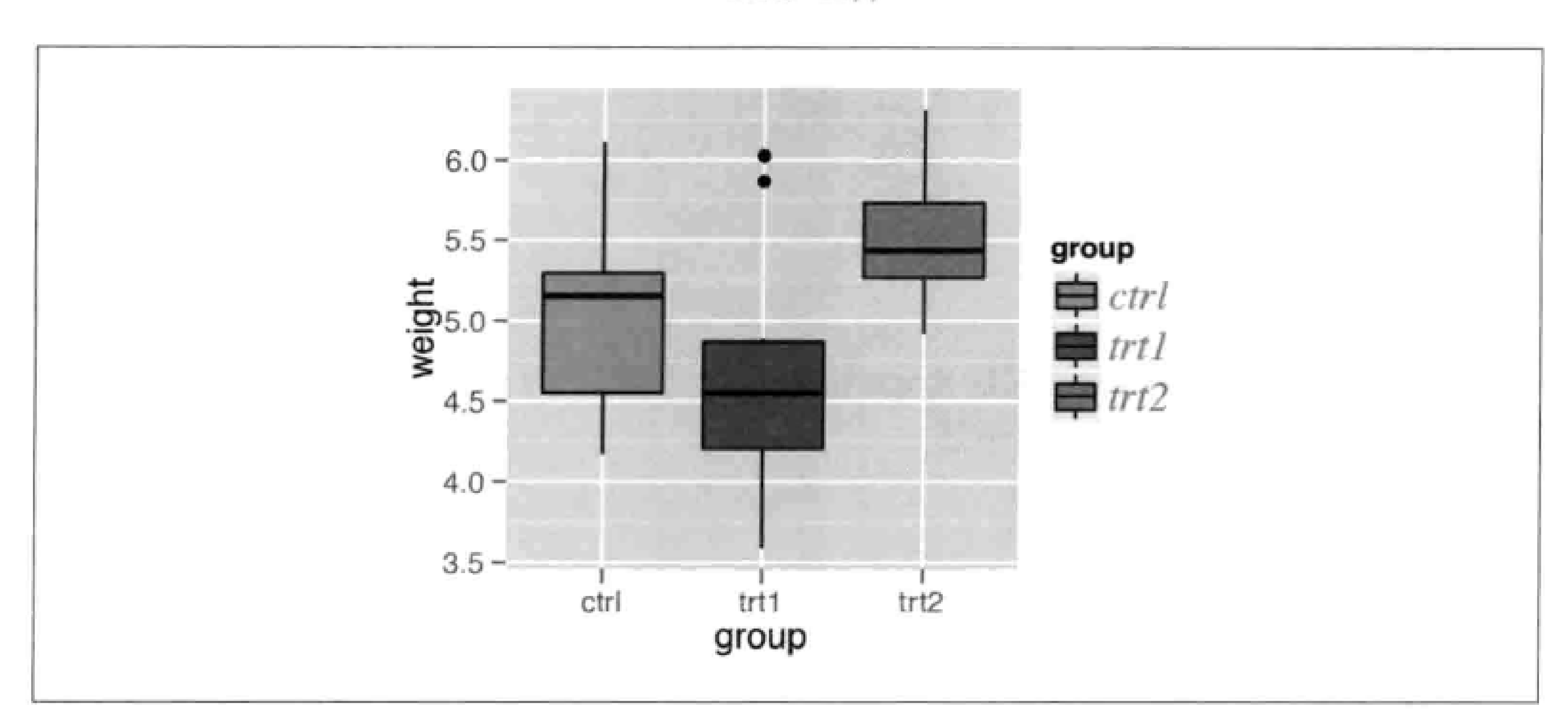


图 10-16　自定义图例标签外观

讨论

我们也可以通过 guides() 来指定图例标签的外观，虽说这种方式有点笨拙。以下代码与上述代码效果相同：

```
# 修改 fill 对应图例标签文本的外观
p + guides(fill=guide_legend(label.theme=
           element_text(face="italic", family="Times", colour="red", size=14)))
```

另见

参见 9.2 节以了解更多关于如何控制文本外观的信息。

10.10 使用含多行文本的标签

问题

如何使用含有多于一行文本的图例标签？

方法

在相应标度中设置 labels 参数，使用 \n 来表示新行。在本例中，我们将使用 scale_fill_discrete() 来控制标度 fill 的图例（见图 10-17 左图）：

```
p <- ggplot(PlantGrowth, aes(x=group, y=weight, fill=group)) + geom_boxplot()

# 含有多于一行文本的标签
p + scale_fill_discrete(labels=c("Control","Type 1\ntreatment",
                                 "Type 2\ntreatment"))
```

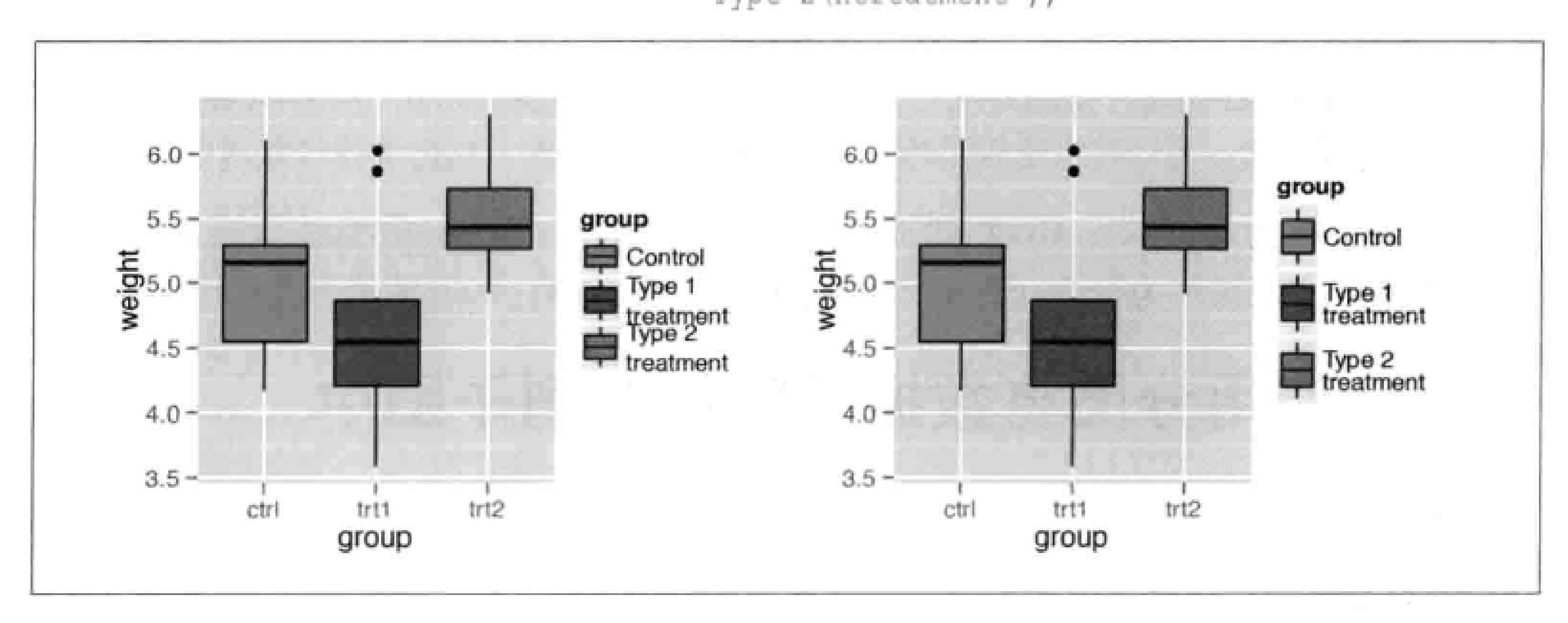


图 10-17　左图：多行的图例标签　右图：增加了图例说明的高度并减小了行距

讨论

从图 10-17 左图可以看到，默认设置下，当使用多于一行文本的标签时，各行文本将相互叠加。要处理这个问题，可以使用 theme() 增加图例说明的高度并减小各行的间距完成（见图 10-17 右图）。要实现这个操作，需要使用 grid 包中的 unit() 函数来指定高度：

```
library(grid)
p + scale_fill_discrete(labels=c("Control","Type 1\ntreatment",
                                 "Type 2\ntreatment")) +
    theme(legend.text=element_text(lineheight=.8),
          legend.key.height=unit(1, "cm"))
```

第11章

分面

数据可视化中最实用的技术之一是将分组数据并列呈现，这样使得组间的比较变得轻而易举。使用 ggplot2 做这件事的方法之一是将一个离散型变量映射为一个图形属性，如 *x* 的位置、颜色或形状。另一种方法则是为每组数据创建一个子图，然后并排绘制这些子图。

这类图形被称为格子（trellis）图形，它们已被 lattice 包和 ggplot2 包所实现。在 ggplot2 中，它们被称为分面（facet）。在本章中，将讲解它们的使用方法。

11.1 使用分面将数据分割绘制到子图中

问题

如何在独立的面板中绘制数据的若干子集？

方法

使用 `facet_grid()` 或 `facet_wrap()` 函数，并指定根据哪个变量来分割数据。

使用 `facet_grid()` 函数时，你可以指定一个变量作为纵向子面板分割的依据，并指定另外一个变量作为横向子面板分割的依据（见图 11-1）：

```
# 基本图形
p <- ggplot(mpg, aes(x=displ, y=hwy)) + geom_point()

# 纵向排列的子面板根据 drv 分面
p + facet_grid(drv ~ .)

# 横向排列的子面板根据 cyl 分面
p + facet_grid(. ~ cyl)

# 同时根据 drv（纵向）和 cyl（横向）分割
p + facet_grid(drv ~ cyl)
```

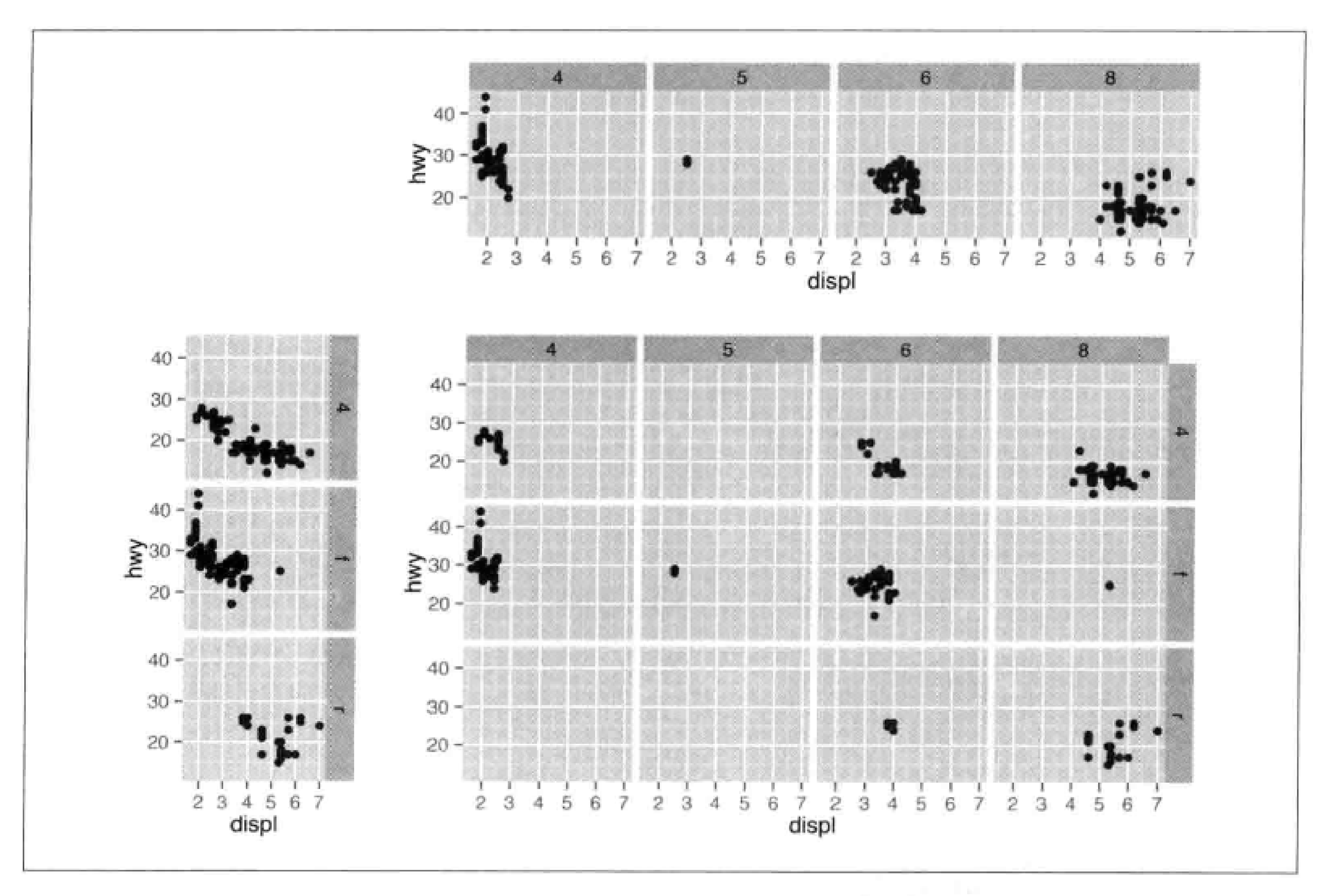


图 11-1　上图：根据 drv 横向分面　左下图：根据 cyl 纵向分面　右下图：使用双变量双向分面

使用 facet_wrap() 时，各子图将像纸上的文字一样被依次横向排布并换行，如图 11-2 所示：

```
# 依 class 分面
# 注意波浪线前没有任何字符
p + facet_wrap( ~ class)
```

讨论

使用 facet_wrap() 时，默认使用相等数量的行和列。图 11-2 中共有 7 个分面，恰好可以嵌入一个 3×3 的“方阵”中。要对该方阵进行修改，可以通过向 nrow 或 ncol 赋值实现：

```
# 两种方式的结果是相同的：2 行 4 列的分面
p + facet_wrap( ~ class, nrow=2)
p + facet_wrap( ~ class, ncol=4)
```

分面方向的选择依赖于你更倾向于鼓励读图者进行哪种类型的比较。举例来说，如果你希望比较各条形的高度，让分面横向排布会更有用；如果你希望比较直方图的水平分布，那么纵向排布分面是明智的选择。

有时两类比较都很重要——所以对于哪种分面方向更好的问题，可能没有一个明确的答案。可能的结果是，通过将分组变量映射到某种如颜色之类的图形属性，在单一的图形中来展示分组，会比使用分面的效果更好。在这些情况下，只能依靠你自己的判断来决定使用哪种方式。

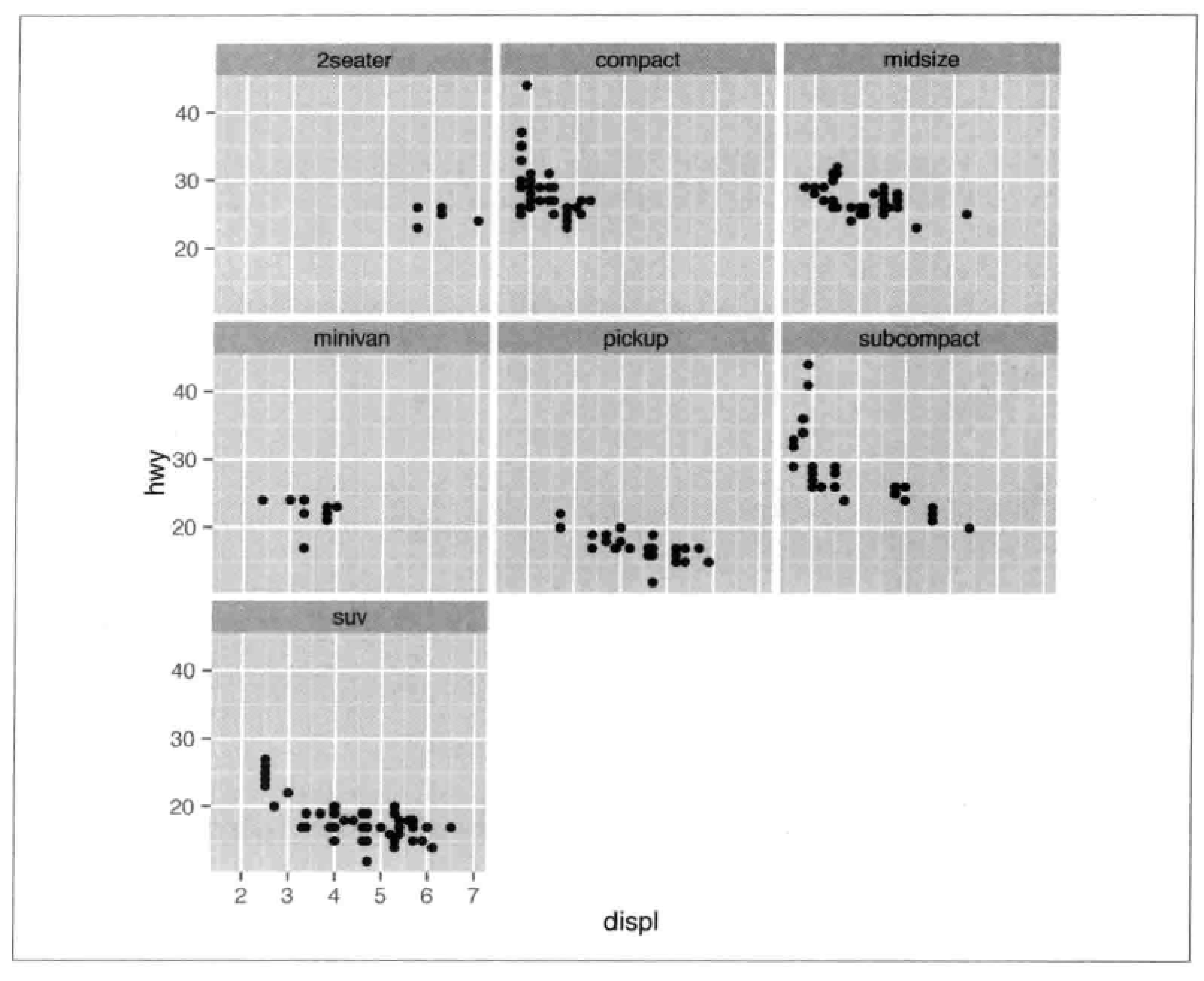


图 11-2 使用 facet_wrap() 对 class 分面得到的散点图

11.2 在不同坐标轴下使用分面

问题

如何绘制坐标轴范围或坐标轴元素不同的多个子图？

方法

将标度设置为 "free_x"、"free_y" 或 "free"（见图 11-3）：

```
# 基本图形
p <- ggplot(mpg, aes(x=displ, y=hwy)) + geom_point()

# 使用自由的 y 标度
p + facet_grid(drv ~ cyl, scales="free_y")

# 使用自由的 x 标度和 y 标度
p + facet_grid(drv ~ cyl, scales="free")
```

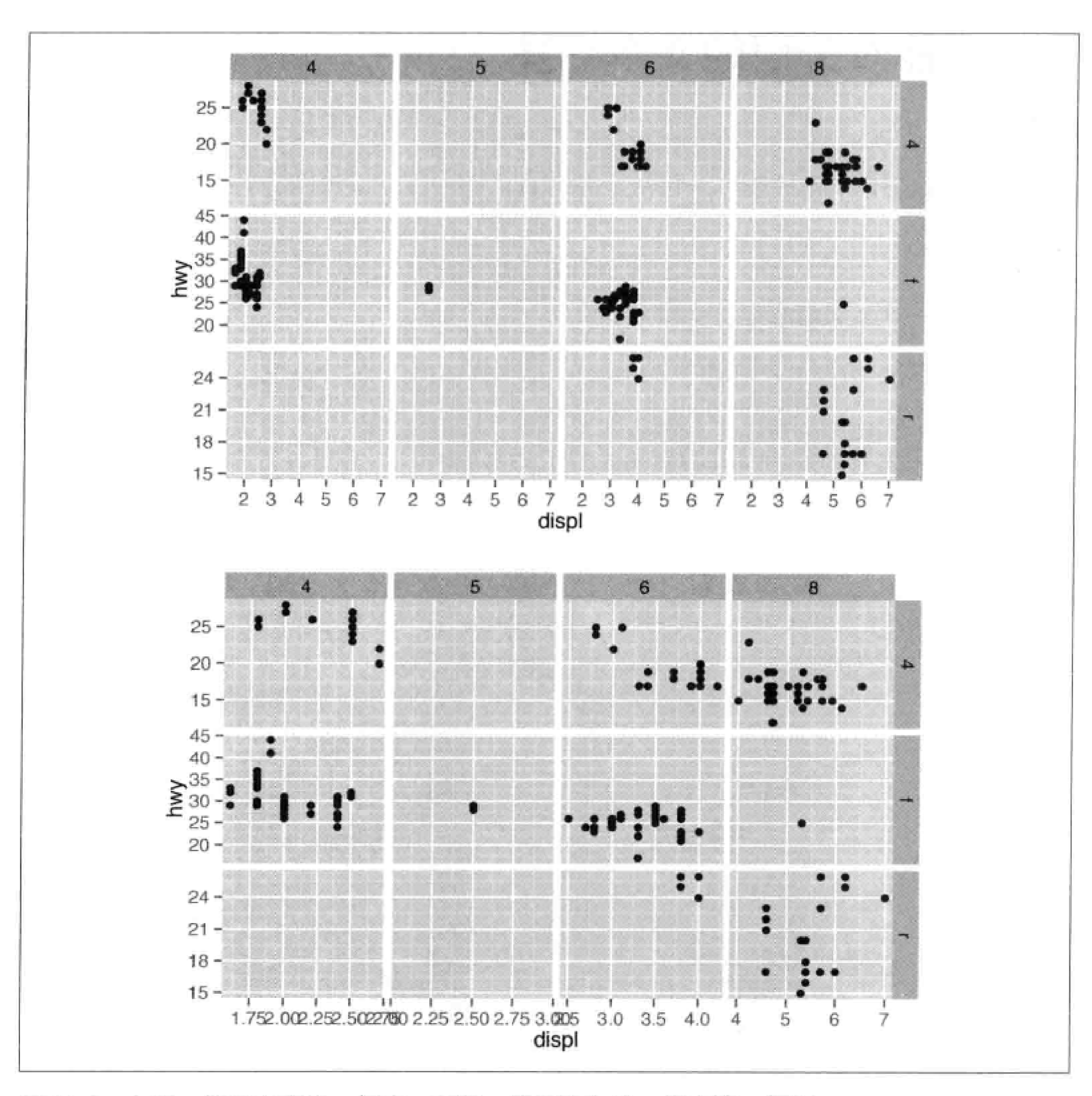


图 11-3　上图：使用自由的 y 标度　下图：使用自由的 x 标度和 y 标度

讨论

当使用自由的 y 标度时，各行子图都将拥有自己的 y 值域；当使用自由的 x 标度时，相同的原理也适用于各列子图。

你无法直接设置各行或各列的值域，但是可以通过丢弃不想要的数据（以缩减值域）或通过添加几何对象 `geom_blank()`（以扩展值域）的方式控制值域的大小。

另见

参见 3.10 节中使用自由标度和离散型坐标轴分面的示例。

11.3 修改分面的文本标签

问题

如何修改分面标签的文本？

方法

修改因子各水平的名称即可（见图 11-4）：

```
mpg2 <- mpg  # 复制一份原始数据

# 重命名 4 为 4wd、f 为 Front、r 为 Rear
levels(mpg2$drv)[levels(mpg2$drv)=="4"] <- "4wd"
levels(mpg2$drv)[levels(mpg2$drv)=="f"] <- "Front"
levels(mpg2$drv)[levels(mpg2$drv)=="r"] <- "Rear"

# 绘制新数据
ggplot(mpg2, aes(x=displ, y=hwy)) + geom_point() + facet_grid(drv ~ .)
```

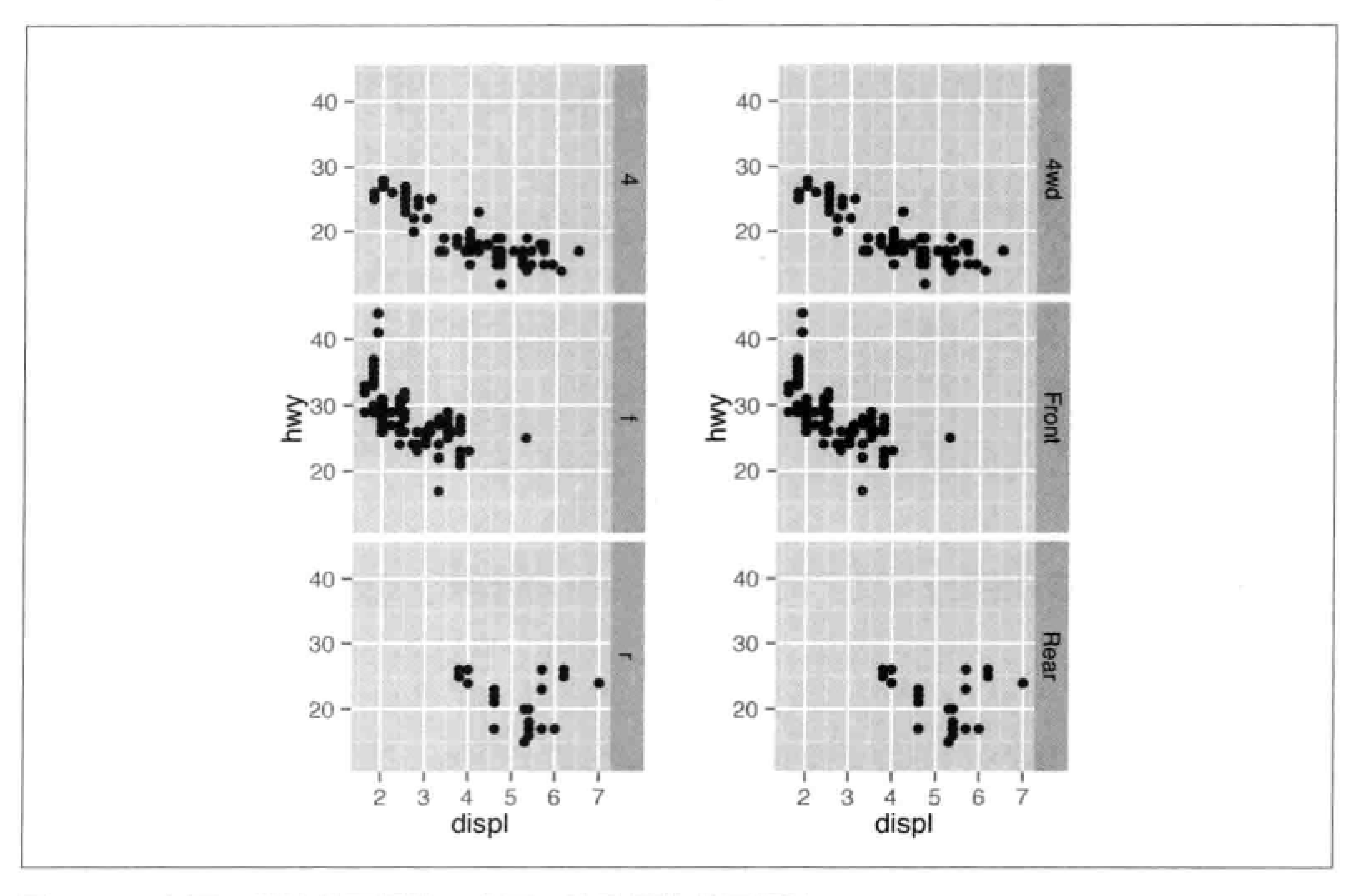


图 11-4　左图：默认分面标签　右图：修改后的分面标签

讨论

要设置分面标签，必须修改数据本身的值，这在用法上与能够设置标签的标度有所不

同。另外，在本书撰写之时，尚无将分面变量的名称作为各分面标题显示的方法，所以使用更有描述力的分面标签是比较有用的。

使用 facet_grid() 时（目前并不适用于 facet_wrap()），可以使用一个贴标函数（labeller function）来设置标签。以下贴标函数 label_both() 将在每个分面上同时打印出变量的名称和变量的值（见图 11-5 左图）：

```
ggplot(mpg2, aes(x=displ, y=hwy)) + geom_point() +
    facet_grid(drv ~ ., labeller = label_both)
```

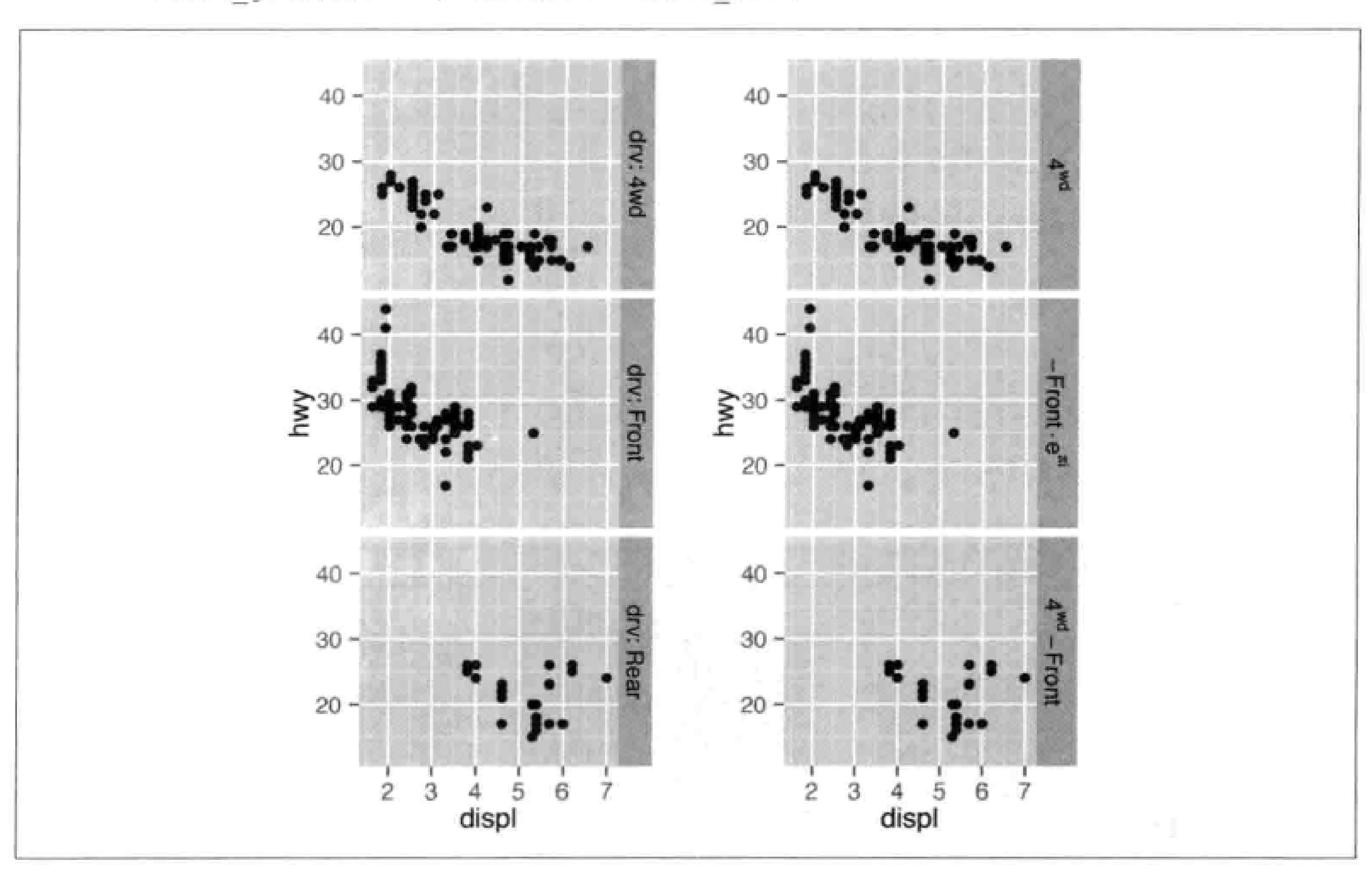


图 11-5　左图：使用 label_both()　右图：使用 label_parsed() 绘制数学表达式

另一个实用贴标函数是 label_parsed()，它可以读入字符串，并将其作为 R 数学表达式来解析（见图 11-5 右图）：

```
mpg3 <- mpg

levels(mpg3$drv)[levels(mpg3$drv)=="4"] <- "4^{wd}"
levels(mpg3$drv)[levels(mpg3$drv)=="f"] <- "- Front %.% e^{pi * i}"
levels(mpg3$drv)[levels(mpg3$drv)=="r"] <- "4^{wd} - Front"

ggplot(mpg3, aes(x=displ, y=hwy)) + geom_point() +
    facet_grid(drv ~ ., labeller = label_parsed)
```

另见

参见 15.10 节以了解更多因子水平重命名的知识。如果分面变量不是一个因子而是一

个字符型向量，修改方式会稍有不同。参考 15.12 节以了解字符型向量元素的重命名。

11.4 修改分面标签和标题的外观

问题

如何修改分面标签和标题的外观？

方法

使用主题系统，通过设置 strip.text 来控制文本的外观，设置 strip.background 以控制背景的外观（见图 11-6）：

```
library(gcookbook)  # 为了使用数据集

ggplot(cabbage_exp, aes(x=Cultivar, y=Weight)) + geom_bar(stat="identity") +
    facet_grid(. ~ Date) +
    theme(strip.text = element_text(face="bold", size=rel(1.5)),
          strip.background = element_rect(fill="lightblue", colour="black",
                                          size=1))
```

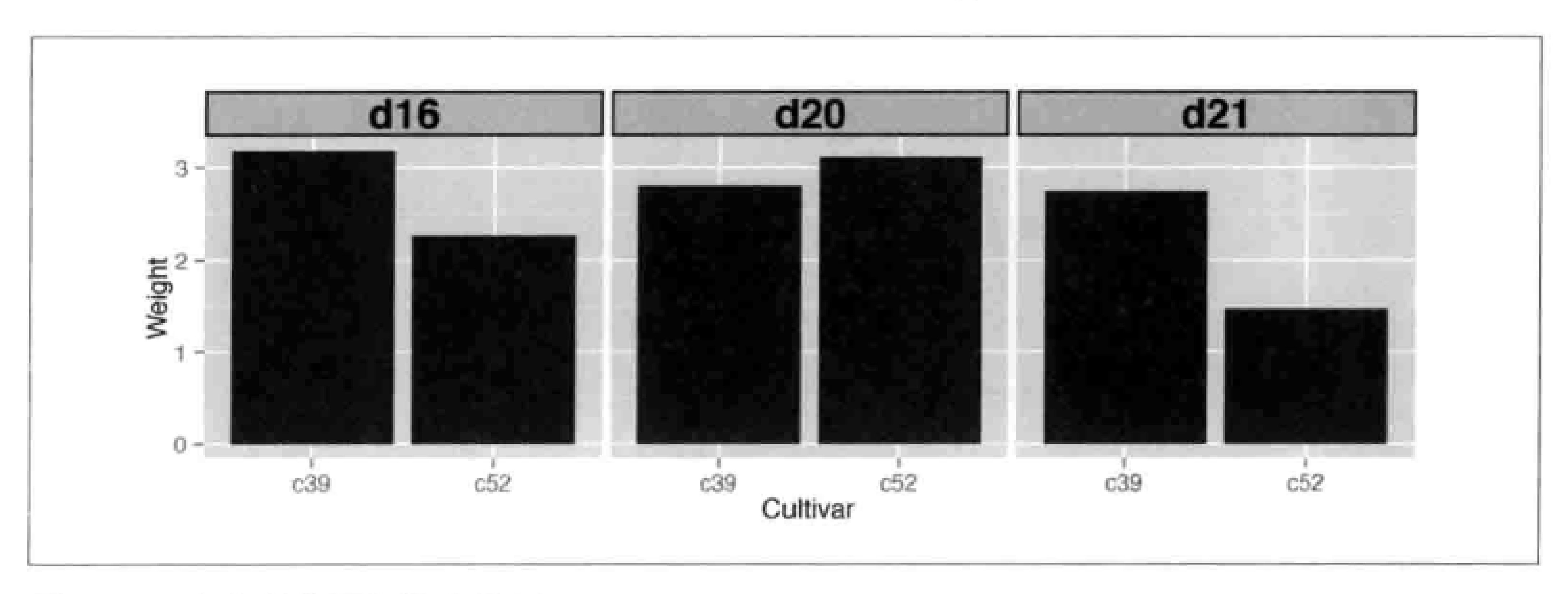


图 11-6 自定义分面标签的外观

讨论

rel(1.5) 使得标签文本的大小为此主题下基准文本大小的 1.5 倍。

在背景设置中使用的 size=1 使得分面标题背景轮廓线的粗细为 1 毫米。

另见

关于主题系统工作原理的更多信息，参见 9.3 节和 9.4 节。

第12章

配色

在 ggplot2 的图形语法中，颜色是一个图形属性（aesthetic），如同 *x* 的位置、*y* 的位置、大小等一样。如果颜色仅仅就是一个简单的图形属性，那又何必单开一章？原因是颜色要比其他图形属性复杂得多。和简单地将几何对象（geom）左右移动、放大放小相比，当你使用颜色的时候，有很多维度或者说自由度要考虑。表达离散型数据时应该用什么样的调色板？是否要使用几种不同的渐变色系？如何选择合适的颜色使得有视觉缺陷的人也能正确地读图？在这一章中，我会解答这些问题。

12.1 设置对象的颜色

问题

如何设置图形中几何对象的颜色？

方法

对于几何对象，设置 `colour` 或者 `fill` 参数的值（见图 12-1）：

```
ggplot(mtcars, aes(x=wt, y=mpg)) + geom_point(colour="red")

library(MASS) # 为了使用数据集
ggplot(birthwt, aes(x=bwt)) + geom_histogram(fill="red", colour="black")
```

讨论

在 ggplot2 中，设置和映射图形属性有非常重大的区别。在前面的例子中，我们将对象的颜色设置为 `"red"`（红色）。

一般而言，`colour` 参数控制的是线条、多边形轮廓的颜色，而 `fill` 参数控制的是多边形的填充色。对于点形来说，情况略微有些不同。大多数的点形，整个点的颜色是由

colour 控制的，而不是 fill。例外的情况是 21-25 号点，它们不仅有填充色，也有边界色。

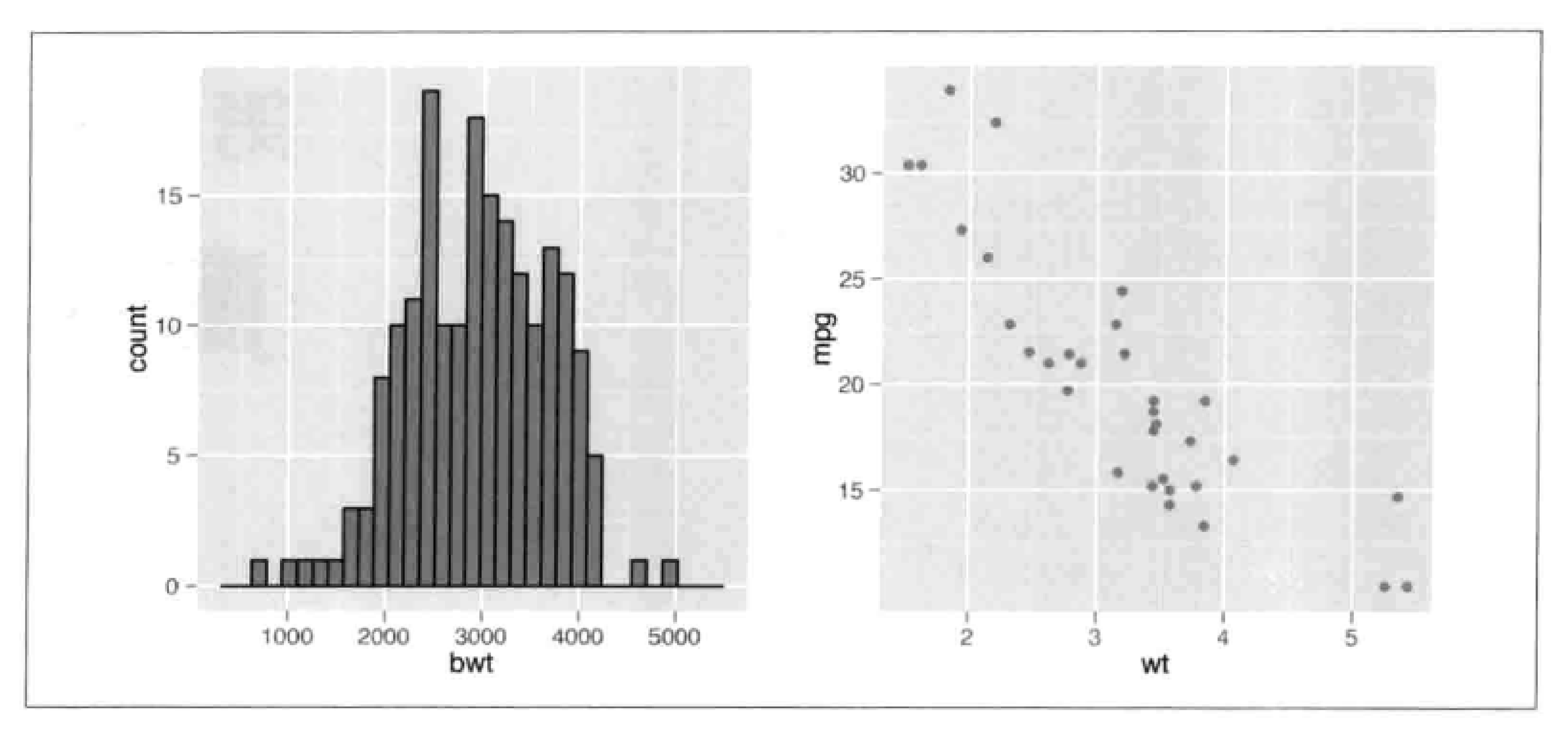


图 12-1　左图：设置填充色和边界色　右图：设置点的颜色

另见

关于不同形状的点的更多信息，参见 4.5 节。

关于设置颜色的更多信息，参见 12.4 节。

12.2　将变量映射到颜色上

问题

如何用一个变量（来自于数据框的某一列）来控制几何对象的颜色？

方法

对于几何对象，将 colour 或 fill 参数的值设置为数据中某一列的列名即可（见图 12-2）：

```
library(gcookbook) # 为了使用数据集

# 这两种方法效果相同
ggplot(cabbage_exp, aes(x=Date, y=Weight, fill=Cultivar)) +
    geom_bar(colour="black", position="dodge")

ggplot(cabbage_exp, aes(x=Date, y=Weight)) +
    geom_bar(aes(fill=Cultivar), colour="black", position="dodge")

# 这两种方法效果相同
ggplot(mtcars, aes(x=wt, y=mpg, colour=cyl)) + geom_point()

ggplot(mtcars, aes(x=wt, y=mpg)) + geom_point(aes(colour=cyl))
```

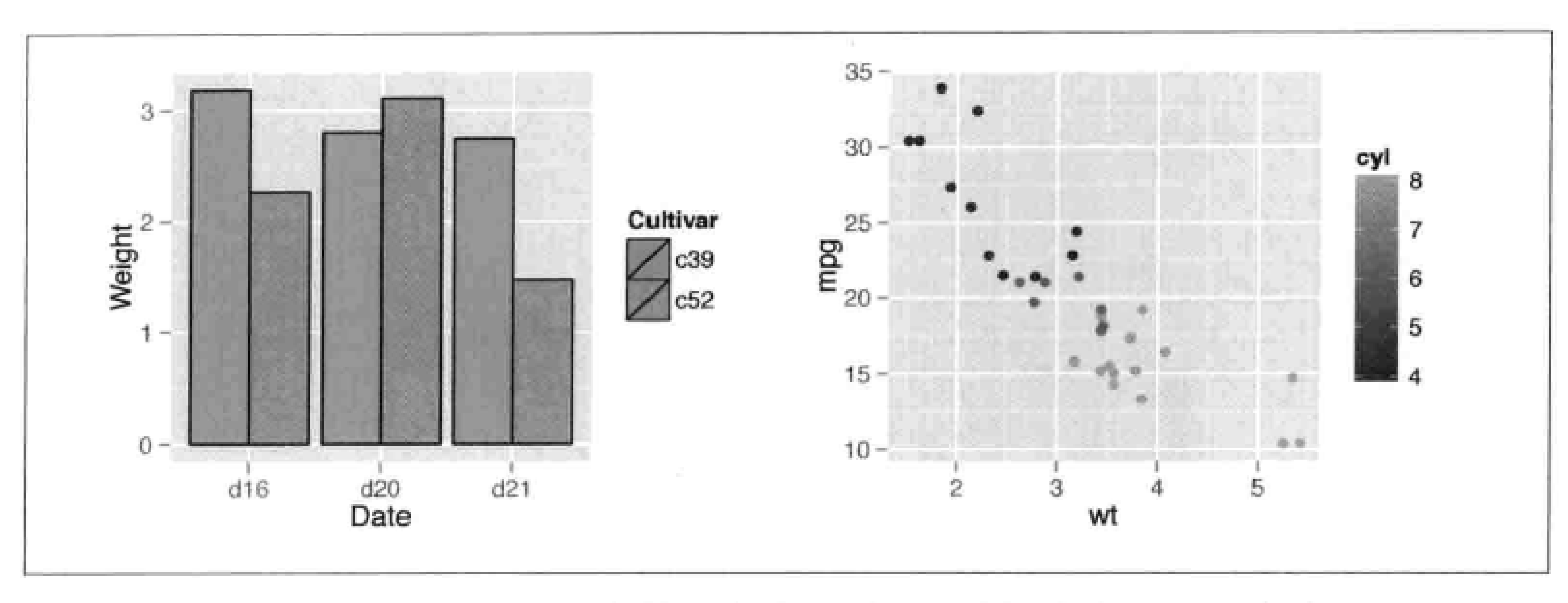


图 12-2　左图：将变量映射到 fill 参数　右图：将变量映射到点的 colour 参数

ggplot() 函数中的映射使用的是默认映射（所有几何对象都由此继承）。通过在 geom 系列函数中的具体设置，可以覆盖默认映射。

讨论

在 cabbage_exp 的例子中，变量 Cultivar 映射到了 fill。cabbage_exp 数据集的列 Cultivar 是因子，因此 ggplot2 将它作为离散型变量处理。你可以输入 str() 来验证：

```
str(cabbage_exp)

'data.frame':   6 obs. of  6 variables:
  $ Cultivar: Factor w/ 2 levels "c39","c52": 1 1 1 2 2 2
  $ Date    : Factor w/ 3 levels "d16","d20","d21": 1 2 3 1 2 3
  $ Weight  : num  3.18 2.8 2.74 2.26 3.11 1.47
  $ sd      : num  0.957 0.279 0.983 0.445 0.791 ...
  $ n       : int  10 10 10 10 10 10
  $ se      : num  0.3025 0.0882 0.311 0.1408 0.2501 ...
```

在 mtcars 的例子中，变量 cyl 是数值形式的，因此它被作为连续型变量来处理。正因为如此，尽管它的实际取值仅仅包含了 4、6、8，图例中还是显示了中间值 5 和 7。为了让 ggplot() 把 cyl 视为分类变量，我们可以在 ggplot() 函数中将其转化为因子，或者修改原数据让需要的列成为字符或因子类型（见图 12-3）。

```
# 在 ggplot() 中因子化
ggplot(mtcars, aes(x=wt, y=mpg, colour=factor(cyl))) + geom_point()

# 另一个方法 : 在原数据中因子化
m <- mtcars                    # 复制 mtcars
m$cyl <- factor(m$cyl)         # 将 cyl 转化为因子
ggplot(m, aes(x=wt, y=mpg, colour=cyl)) + geom_point()
```

另见

你可能还需要改变在标度中使用的颜色。对于连续型数据，参见 12.6 节；对于离散型

数据，参见 12.3 节或 12.4 节。

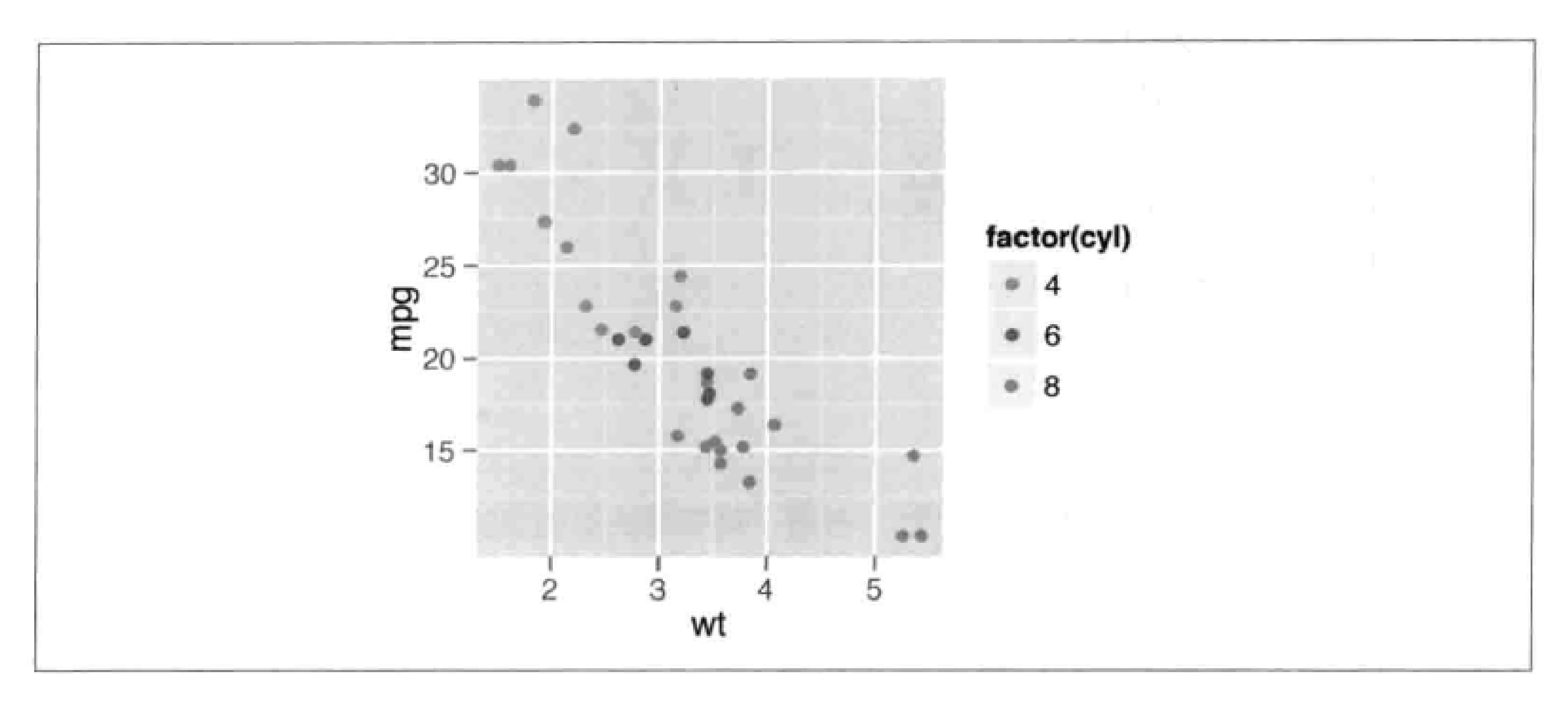


图 12-3　将连续型变量转换为因子映射到颜色上

12.3　对离散型变量使用不同的调色板

问题

如何对离散映射的变量使用不同的颜色？

方法

使用表 12-1 中列出的一种标度。

表 12-1　离散的填充色标度和轮廓色标度

填充色标度	轮廓色标度	描述
scale_fill_discrete()	scale_colour_discrete()	色轮周围均匀等距色（同 hue）
scale_fill_hue()	scale_colour_hue()	色轮周围均匀等距色（同 discrete）
scale_fill_grey()	scale_colour_grey()	灰度调色板
scale_fill_brewer()	scale_colour_brewer()	ColorBrewer 调色板
scale_fill_manual()	scale_colour_manual()	自定义颜色

在本例中，我们将会使用默认调色板和 ColorBrewer 调色板（见图 12-4）：

```
library(gcookbook) # 为了使用数据集

# 基础图形
p <- ggplot(uspopage, aes(x=Year, y=Thousands, fill=AgeGroup)) + geom_area()
```

```
# 这三种方法效果相同
p
p + scale_fill_discrete()
p + scale_fill_hue()

# ColorBrewer 调色板
p + scale_fill_brewer()
```

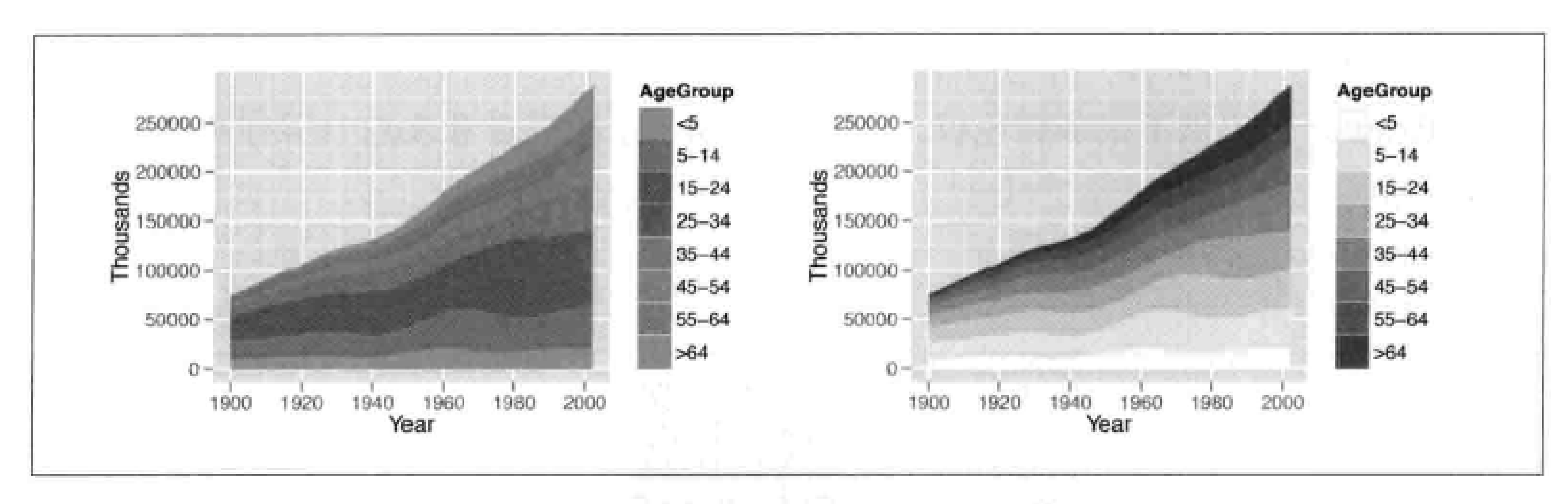


图 12-4 左图：默认调色板（hue） 右图：一个 ColorBrewer 调色板

讨论

修改调色板就是修改填充色标度（fill）或轮廓色标度（colour）：它会涉及从连续型或离散型变量到图形属性上映射的改变。在颜色上有两个类型的标度，填充色标度和轮廓色标度。

函数 scale_fill_hue() 中，颜色来自 HCL 色系（hue-chroma-lightness：色相 - 色度 - 亮度）的色轮，默认的亮度是 65（取值为 0~100）。这很适合作为填充色，但对点、线条来说略微亮了些。为了使点、线条的颜色更深一点（见图 12-5 右图），可以设置亮度参数 l 值（luminance/lightness）来实现。

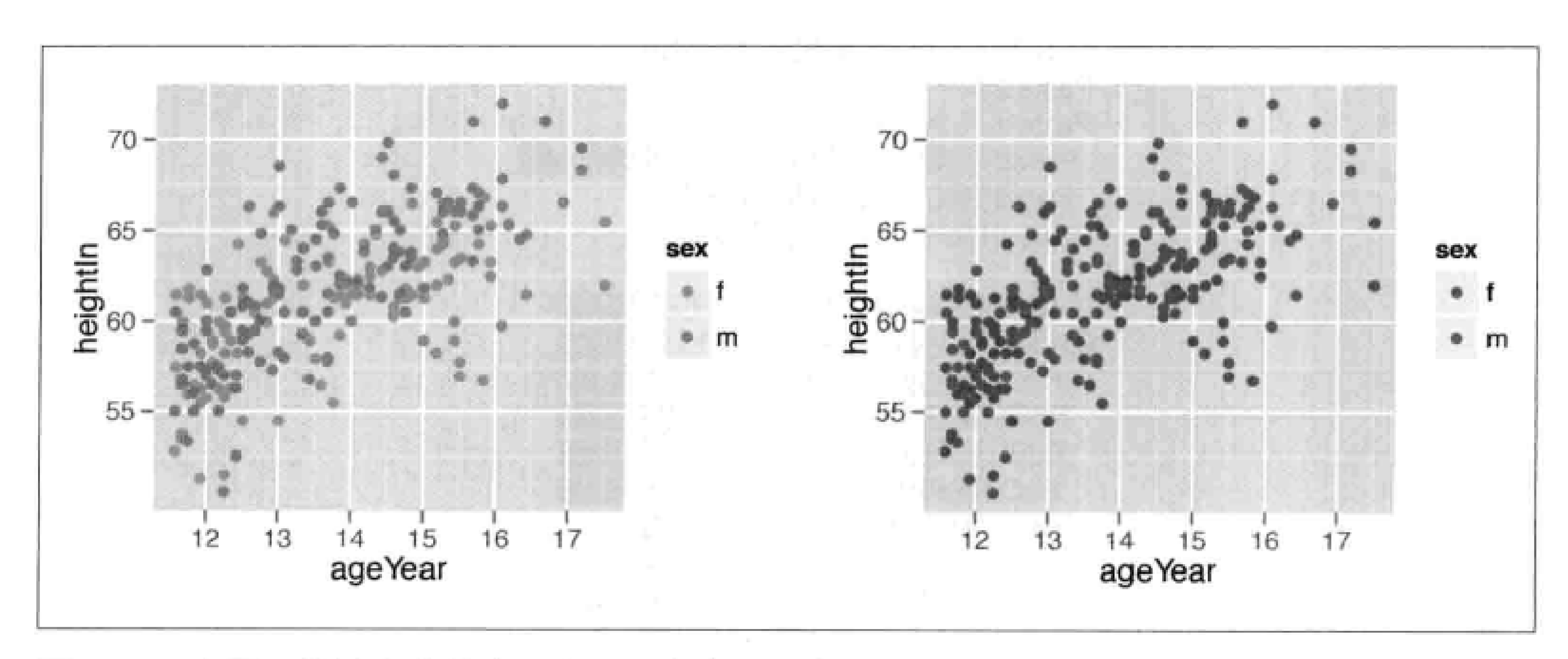


图 12-5 左图：默认亮度的点 右图：亮度设置为 45

```
# 基本的散点图
h <- ggplot(heightweight, aes(x=ageYear, y=heightIn, colour=sex)) +
     geom_point()

# 默认亮度 lightness = 65
h

# 略微加深
h + scale_colour_hue(l=45)
```

ColorBrewer 包提供了很多调色板。你可以生成一张图来查看该包中所有调色板，参见图 12-6：

```
library(RColorBrewer)
display.brewer.all()
```

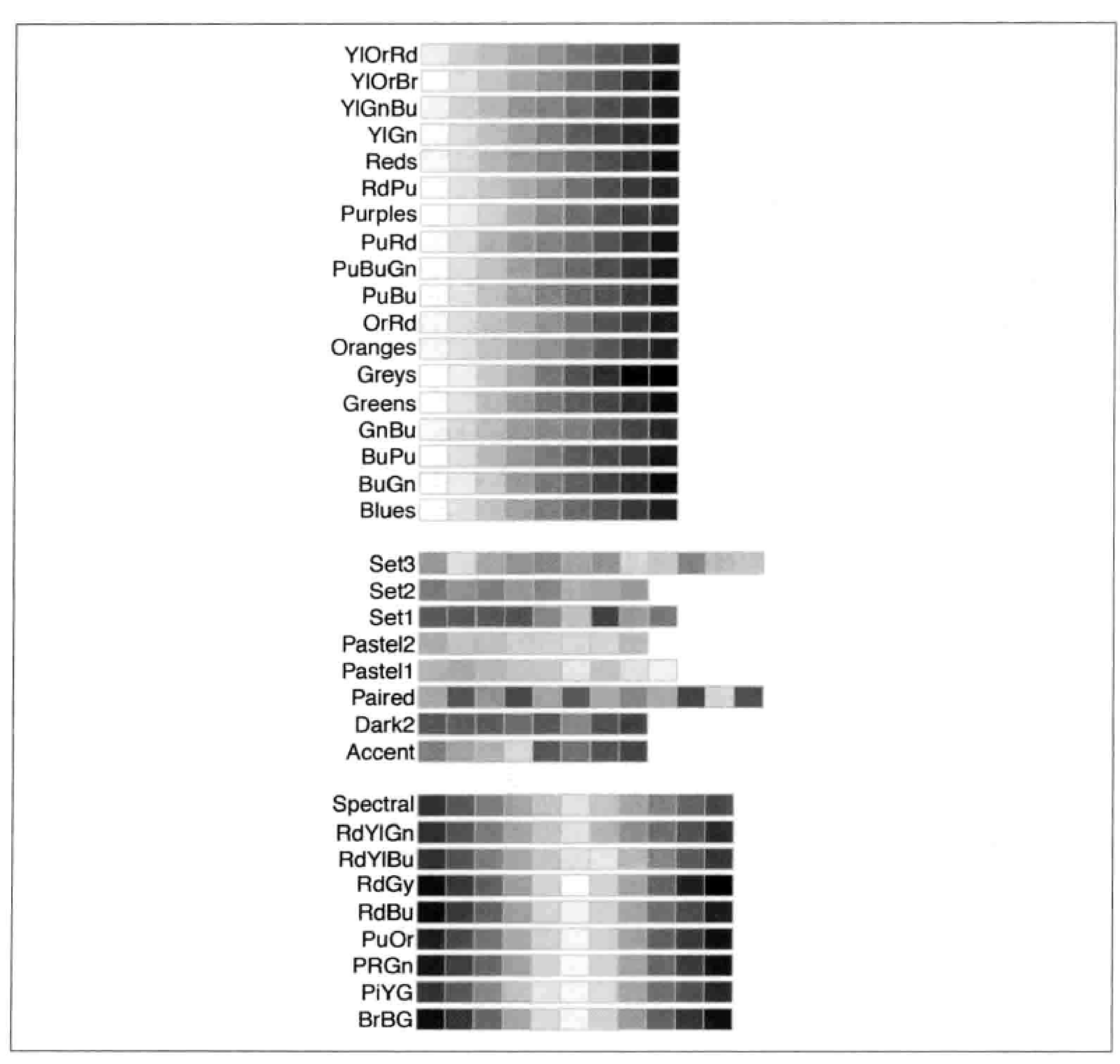


图 12-6　所有的 ColorBrewer 调色板

ColorBrewer 调色板可以通过名称来选择。比如，这将使用橘黄色调色板（见图 12-7）：

```
p + scale_fill_brewer(palette="Oranges")
```

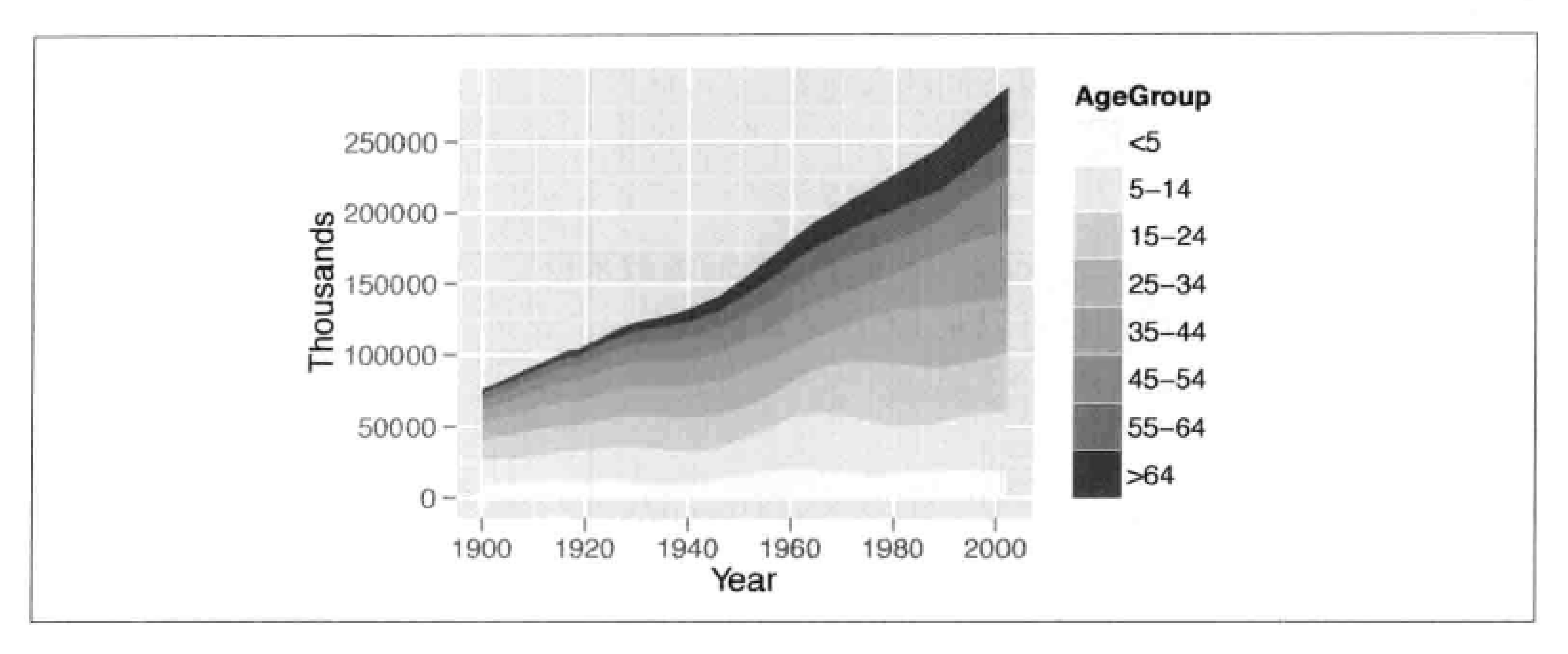


图 12-7 使用橘黄色的 ColorBrewer 调色板

你还可以使用灰度调色板，它很适合黑白打印。标度范围是 0 ～ 1（其中 0 对应黑色，1 对应白色），灰度调色板的默认范围是 0.2 ～ 0.8，但这个可以更改，参见图 12-8。

```
p + scale_fill_grey()

# 倒转方向并且更改灰度范围
p + scale_fill_grey(start=0.7, end=0)
```

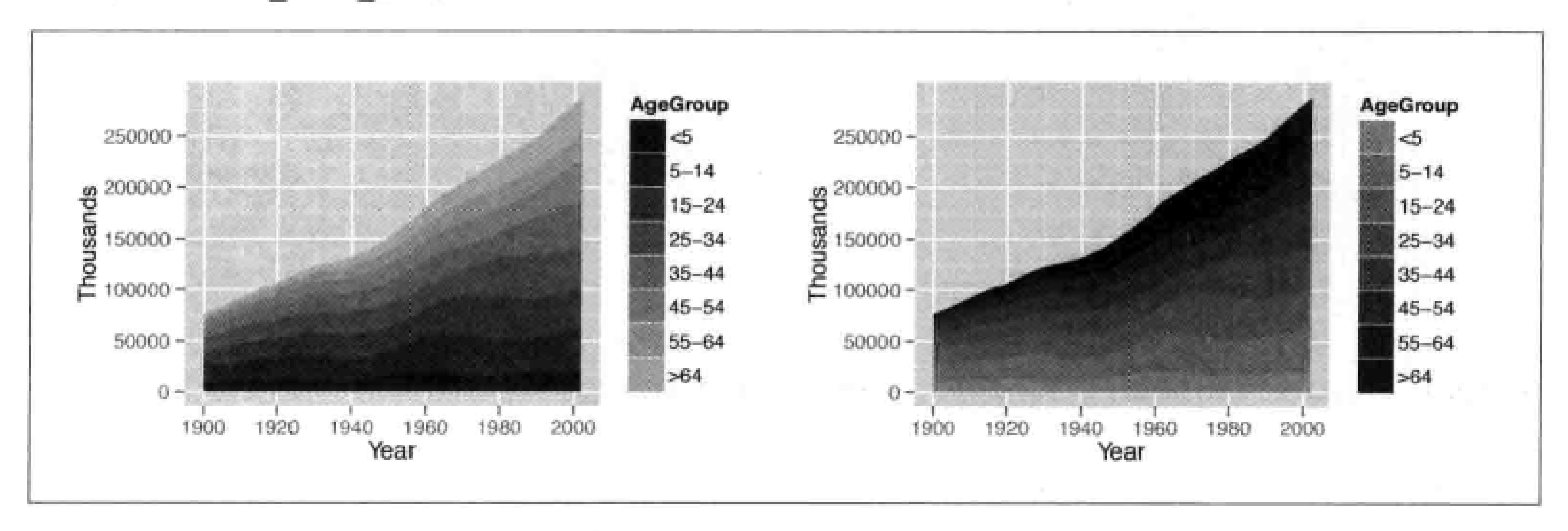


图 12-8 左图：使用默认的灰度调色板 右图：一个不同的灰度调色板

另见

10.4 节中讲述了倒转图例。

手动选择颜色，参见 12.4 节。

关于 ColorBrewer 的更多信息参见 http://colorbrewer.org。

12.4 对离散型变量使用自定义调色板

问题

对离散映射的变量，如何使用不同的颜色？

方法

在本例中，我们将用 `scale_colour_manual()` 函数来自定义颜色（见图 12-9）。其中的颜色可以是已命名的，也可以是 RGB 形式的。

```
library(gcookbook) # 为了使用数据集

# 基础图形
h <- ggplot(heightweight, aes(x=ageYear, y=heightIn, colour=sex)) + geom_point()

# 使用颜色名
h + scale_colour_manual(values=c("red", "blue"))

# 使用 RGB 值
h + scale_colour_manual(values=c("#CC6666", "#7777DD"))
```

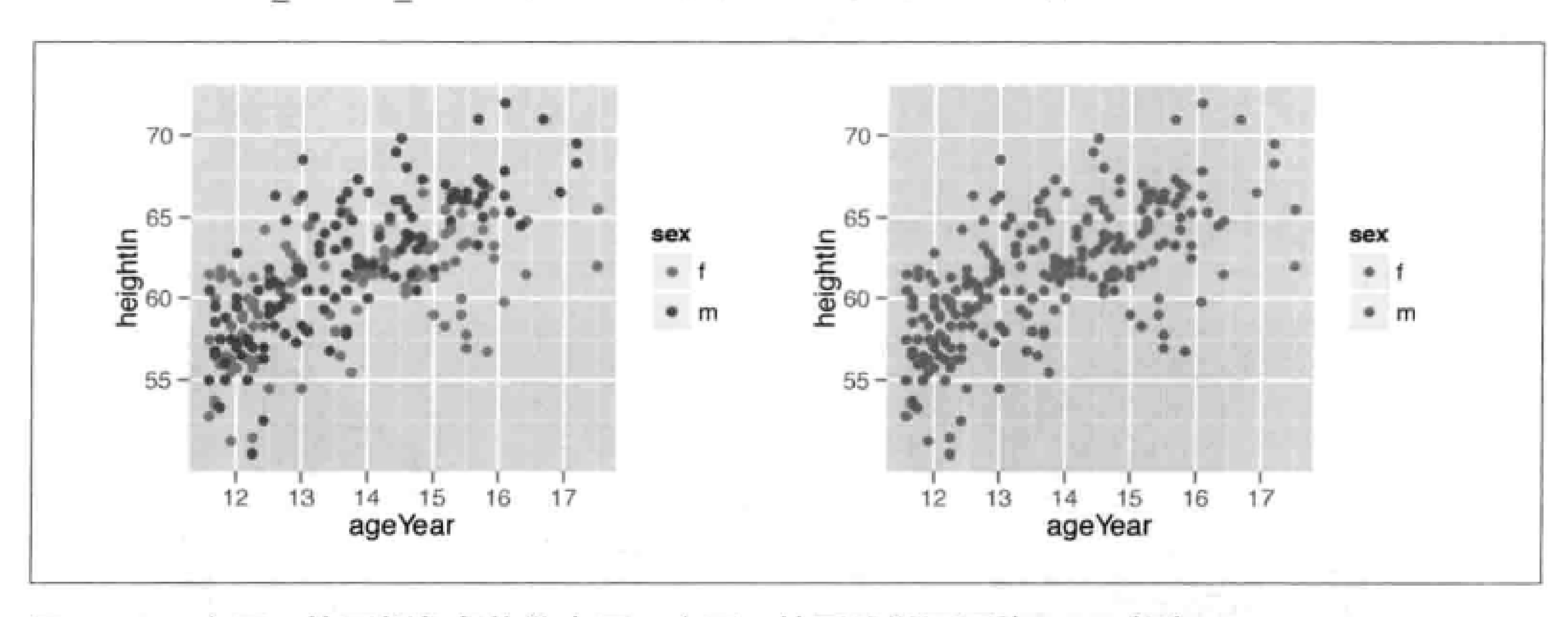


图 12-9　左图：使用颜色名的散点图　右图：使用略微不同的 RGB 颜色

对于填充色标度，使用 `scale_fill_manual()` 代替即可。

讨论

参数 `values` 向量中的元素顺序自动匹配离散标度对应因子水平的顺序。在前面的例子中，`sex` 的顺序是先 `f` 后 `m`，因此 `values` 的第一个值赋予 `f`，第二个值赋予 `m`。下面是如何查看因子顺序的方法：

```
levels(heightweight$sex)

 "f" "m"
```

如果变量是字符型向量而非因子形式，那么它会被自动转化为因子；顺序也默认地按字母表排序。

如果想自定义颜色分配的顺序，可以使用带有名称的向量参数：

```
h + scale_colour_manual(values=c(m="blue", f="red"))
```

在 R 中有很多已经命名的颜色，你可以运行 colors() 命令来查看。知道一些基本的颜色名是很有用的，比如 "white"、"black"、"grey80"、"red"、"blue"、"darkred" 等。还有很多其他的颜色，但是它们的名字并不是很好理解（比如我就对 "thistle3"、"seashell" 之类的毫无概念），因此使用 RGB 值来定义颜色往往更容易些。

RGB 颜色是由六个数字构成（十六进制数），形式如 "#RRGGBB"。在十六进制中，数字先从 0 到 9，然后紧接着是 A 到 F，其中 A 对应十进制中的 10，F 对应十进制中的 15。每一个颜色都由两个数字来表示，范围从 00 到 FF（(对应十进制中的 255)。举个例子，颜色 "#FF0099" 中，255 表示红色、0 表示绿色、153 表示蓝色，整体表示品红色。十六进制数中每个颜色通道常常重复同样的数字，因为这样更容易阅读并且第二个数字的精确值对外观的影响并不是很明显。

这里总结了一些设置、调整 RGB 颜色的经验法则。

- 在一般情况下，较大的数字更明亮，较小的数字更暗淡。
- 如果要得到灰色，将三个颜色通道设置为相同的值。
- 和 RGB 对立的是 CMY（印刷三原色）：青（cyan）、品红（magenta）、黄（yellow）。红色通道值越高，颜色越红，越低则越青。同样的法则适用于另外两组颜色对：绿色 - 品红、蓝色 - 黄色。

另见

这里提供了一个 RGB 颜色十六进制表：http://html-color-codes.com/。

12.5 使用色盲友好式的调色板

问题

如何选择色盲朋友也能正确区分的颜色？

方法

使用函数 scale_fill_manual()，调色板（cb_palette）用自定义的（见图 12-10）：

```
library(gcookbook) # 为了使用数据集

# 基础图形
p <- ggplot(uspopage, aes(x=Year, y=Thousands, fill=AgeGroup)) + geom_area()
```

```
# 加入灰色到调色板：
cb_palette <- c("#999999", "#E69F00", "#56B4E9", "#009E73", "#F0E442",
                "#0072B2", "#D55E00", "#CC79A7")

# 将其使用到图形中
p + scale_fill_manual(values=cb_palette)
```

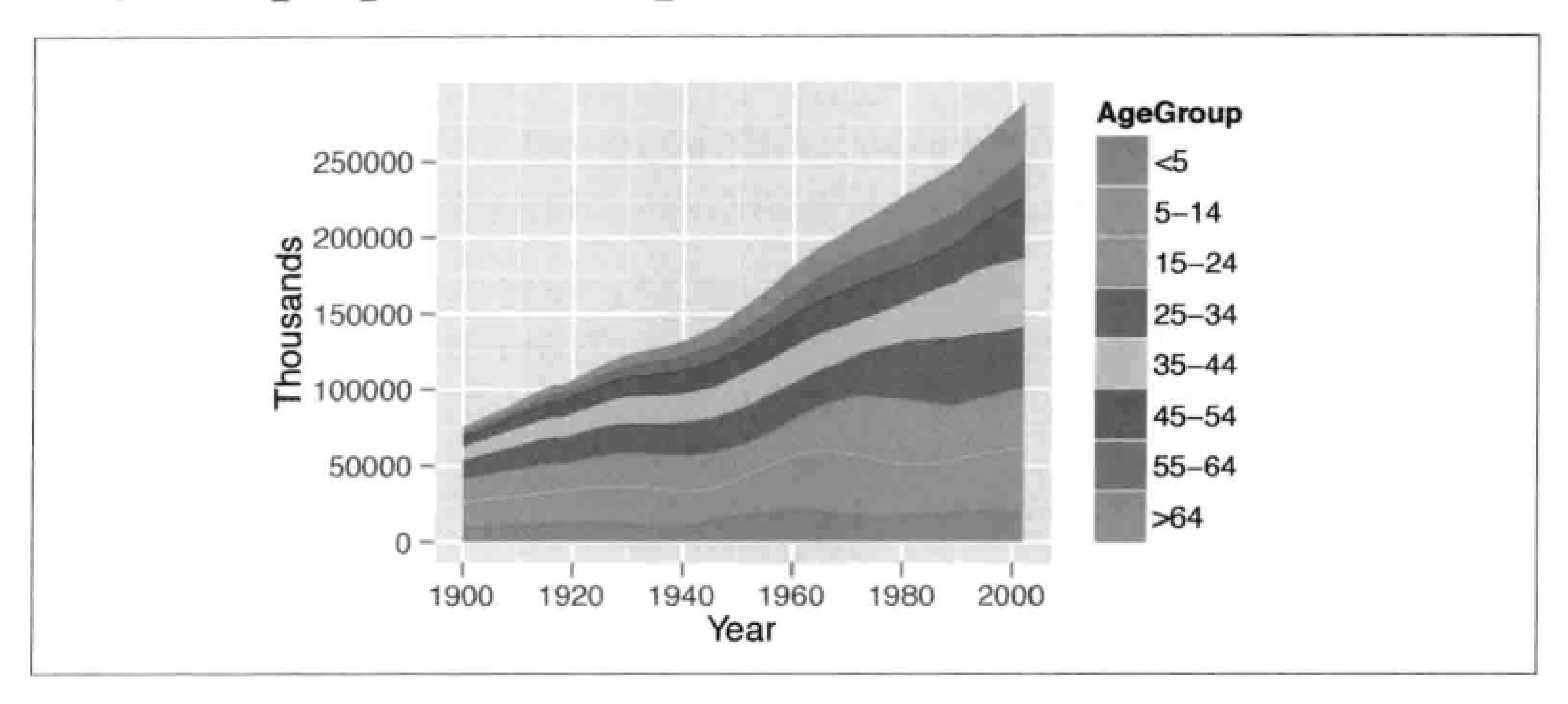


图 12-10　一个色盲友好式的图形

图 12-11 中展示了颜色。

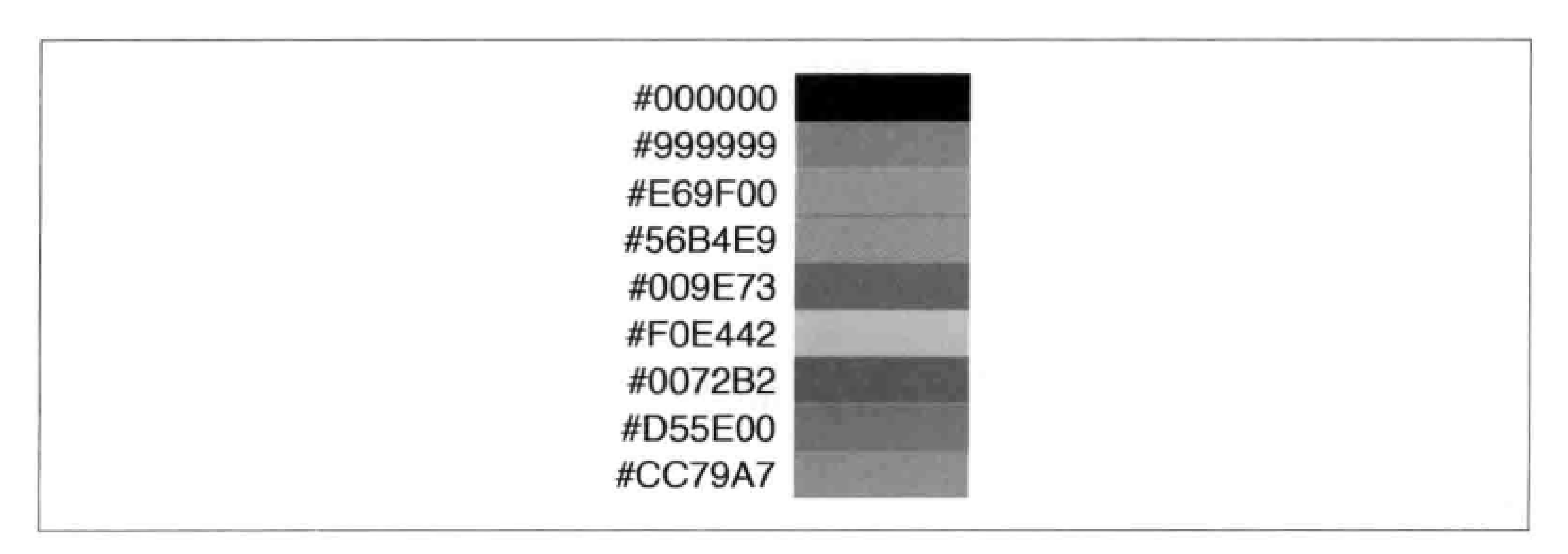


图 12-11　RGB 数值的色盲调色板

有时候使用黑色可能比灰色更好，此时将 "#999999" 替换为 "#000000" 或 "black" 即可：

```
c("#000000", "#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2", "#D55E00",
  "#CC79A7")
```

讨论

大约 8% 的男性、0.5% 的女性存在某种形式的颜色识别缺陷，因此你的周围很可能就有这样的人。

色盲的形式多种多样，这里的调色板是经专门设计、能让所有常见的色觉缺陷人士区分的（Monochromacy，或全色盲，是非常罕见的。这种色盲人群只能区分亮度的差异）。

另见

本调色板的来源：http://jfly.iam.u-tokyo.ac.jp/color/。

Color Oracle 程序（http://colororacle.org/）可以仿真模拟出你屏幕上的东西在色觉缺陷人士眼中的样子，不过要记住的是这个仿真模拟并不完善。在我非正式的测试中，我看了一幅模拟红绿色缺陷的图片，可以很好地区分，但真正的红绿色盲却无法区分！

12.6 对连续型变量使用自定义调色板

问题

如何对连续型变量自定义调色板？

方法

在本例中，我们将会展示对连续型变量自定义渐变式的调色板（见图 12-12）。颜色可以用已命名的，也可以用 RGB 值来指定。

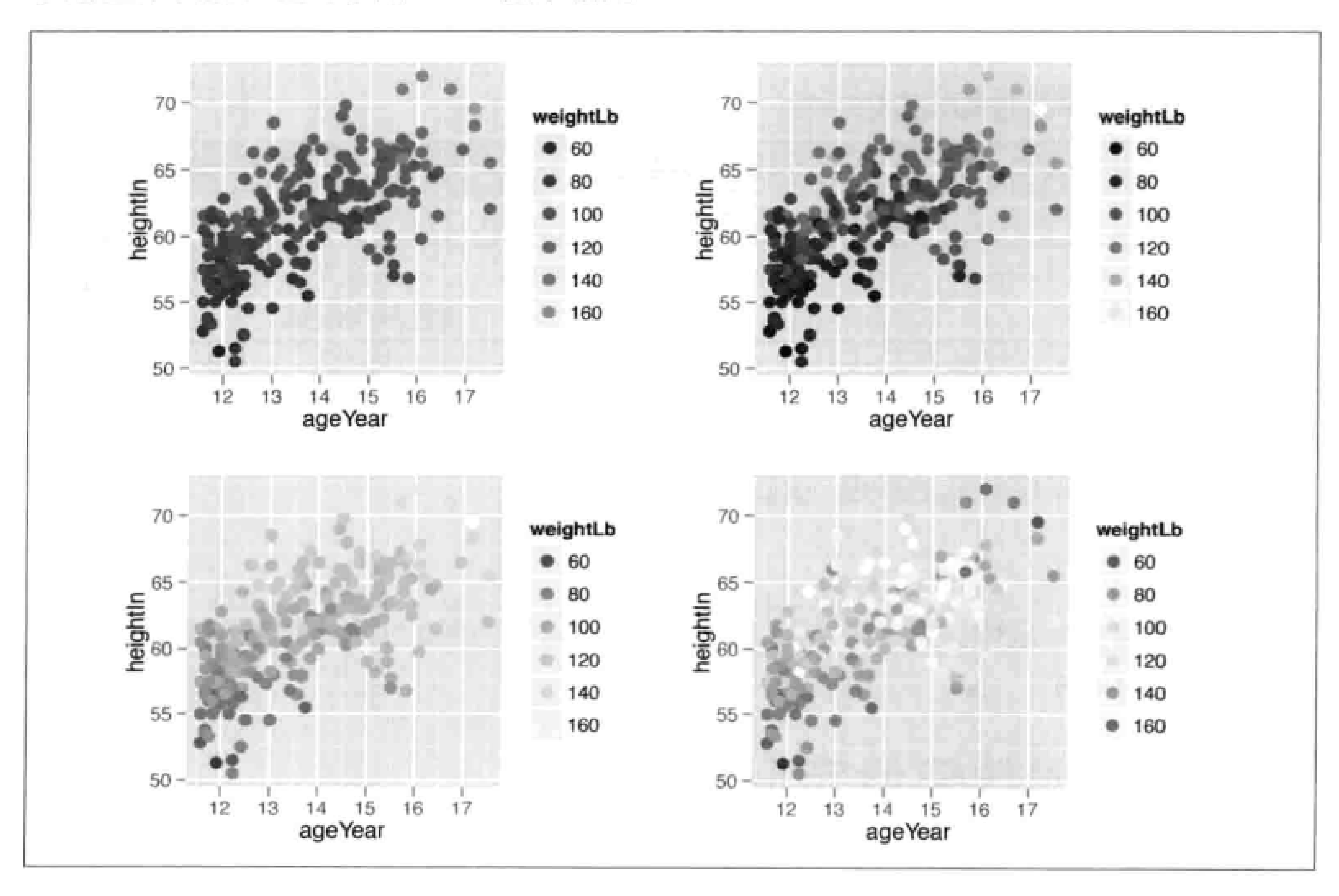


图12-12 从左上图顺时针依次是：默认颜色、两色渐变（scale_colour_gradient()）、三色渐变（scale_colour_gradient2()）和四色渐变（scale_colour_gradientn()）

```
library(gcookbook) # 为了使用数据集

# 基础图形
p <- ggplot(heightweight, aes(x=ageYear, y=heightIn, colour=weightLb)) +
     geom_point(size=3)
p

# 使用两种颜色的渐变色
p + scale_colour_gradient(low="black", high="white")

# 渐变色中间用白色划分
library(scales)
p + scale_colour_gradient2(low=muted("red"), mid="white", high=muted("blue"),
    midpoint=110)

# n个颜色的渐变色
p + scale_colour_gradientn(colours = c("darkred", "orange", "yellow", "white"))
```

对于填充色标度，使用 scale_fill_xxx() 替换即可，这里 xxx 是 gradient、gradient2 或 gradientn 中的一个。

讨论

将连续型变量映射到颜色标度上需要一组连续变化的颜色。表 12-2 列出了连续的填充色和轮廓色标度。

表 12-2　连续的填充色和轮廓色标度

填充色标度	轮廓色标度	描　述
scale_fill_gradient()	scale_colour_gradient()	两色渐变
scale_fill_gradient2()	scale_colour_gradient2()	三色渐变，由中间色、两端色渐变组成
scale_fill_gradientn()	scale_colour_gradientn()	等间隔的 *n* 种颜色的渐变色

请注意，我们在本例中使用了 muted() 函数。该函数来自 scales 包，它会针对输入的颜色输出一个饱和度较低的颜色（RGB 格式）。

另见

如果你不想用连续标度，而想使用离散（分类）标度，你可以将数据重新编码成分类值。参见 15.14 节。

12.7　根据数值设定阴影颜色

问题

如何根据 *y* 值设置阴影的颜色？

方法

增加一列来对 y 进行划分，然后将该列映射到填充色标度上。在本例中，首先对数据进行正负划分。

```
library(gcookbook) # 为了使用数据集

cb <- subset(climate, Source=="Berkeley")

cb$valence[cb$Anomaly10y >= 0] <- "pos"
cb$valence[cb$Anomaly10y < 0]  <- "neg"

cb
```

```
   Source Year Anomaly1y Anomaly5y Anomaly10y Unc10y valence
 Berkeley 1800        NA        NA     -0.435  0.505     neg
 Berkeley 1801        NA        NA     -0.453  0.493     neg
 Berkeley 1802        NA        NA     -0.460  0.486     neg
 ...
 Berkeley 2002        NA        NA      0.856  0.028     pos
 Berkeley 2003        NA        NA      0.869  0.028     pos
 Berkeley 2004        NA        NA      0.884  0.029     pos
```

当我们对数据划分正负之后，我们就可以将 valence 变量映射到填充色上来作图了，如图 12-13 所示：

```
ggplot(cb, aes(x=Year, y=Anomaly10y)) +
    geom_area(aes(fill=valence)) +
    geom_line() +
    geom_hline(yintercept=0)
```

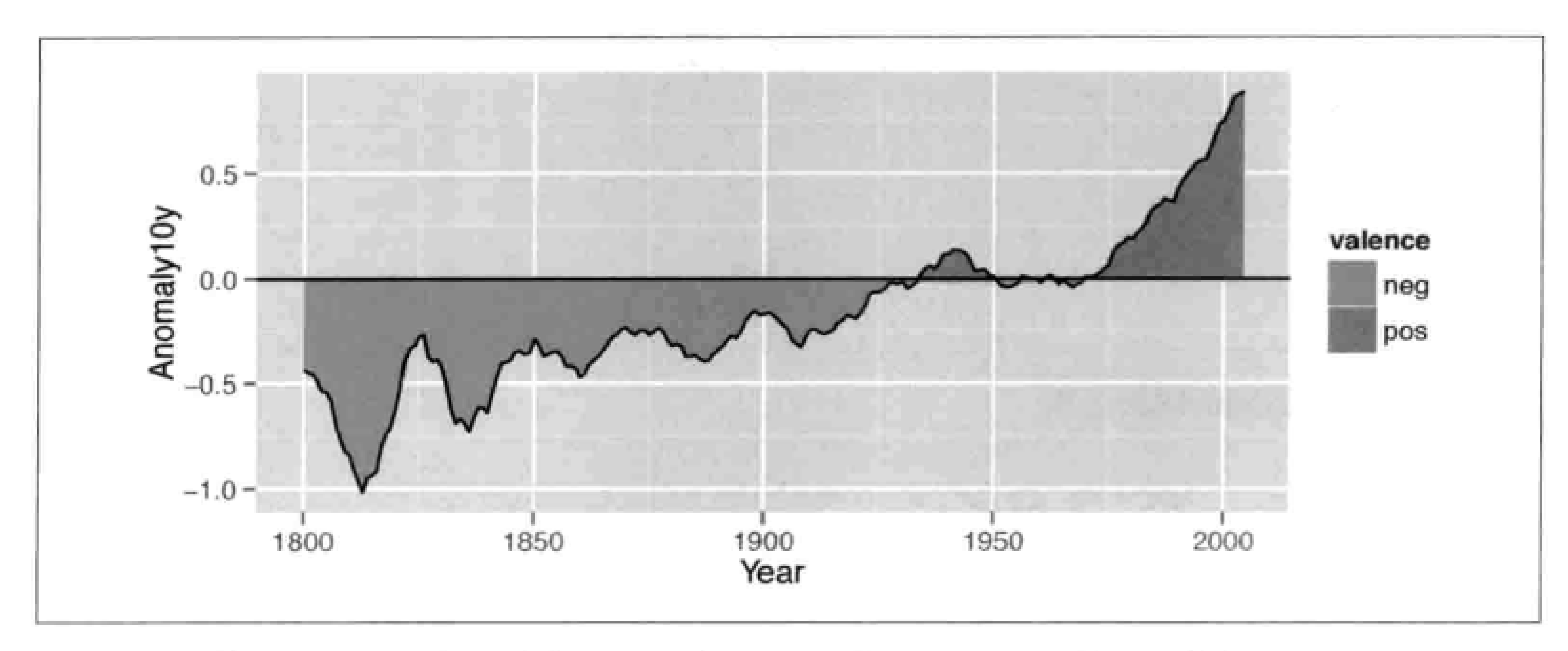


图 12-13　将 valence 映射到填充色——注意 1950 年附近 0 水平线下面的红色区域

讨论

如果你仔细观察此图，会发现在 0 水平线附近有一些凌乱的阴影。这是因为两个颜色

区域都是由各自的数据点多边形包围形成的，而这些数据点并不都在 0 上。为了解决这个问题，我们可以用 approx() 将数据插值到 1000 个点左右。

```
# approx() 返回一个列表，包含 x 和 y 向量
interp <- approx(cb$Year, cb$Anomaly10y, n=1000)

# 放在一个数据框中并重新计算 valence
cbi <- data.frame(Year=interp$x, Anomaly10y=interp$y)
cbi$valence[cbi$Anomaly10y >= 0] <- "pos"
cbi$valence[cbi$Anomaly10y < 0]  <- "neg"
```

我们也可以让插值精确地通过零点，但显然这样会更复杂。在这里，approx() 的效果已经很好了。

现在，我们重新绘制插值后的数据（见图 12-14）。这一次，我们还会做一些调整——让阴影区域半透明、改变颜色、移除图例并删除填充区域左右两侧的空余：

```
ggplot(cbi, aes(x=Year, y=Anomaly10y)) +
    geom_area(aes(fill=valence), alpha = .4) +
    geom_line() +
    geom_hline(yintercept=0) +
    scale_fill_manual(values=c("#CCEEFF", "#FFDDDD"), guide=FALSE) +
    scale_x_continuous(expand=c(0, 0))
```

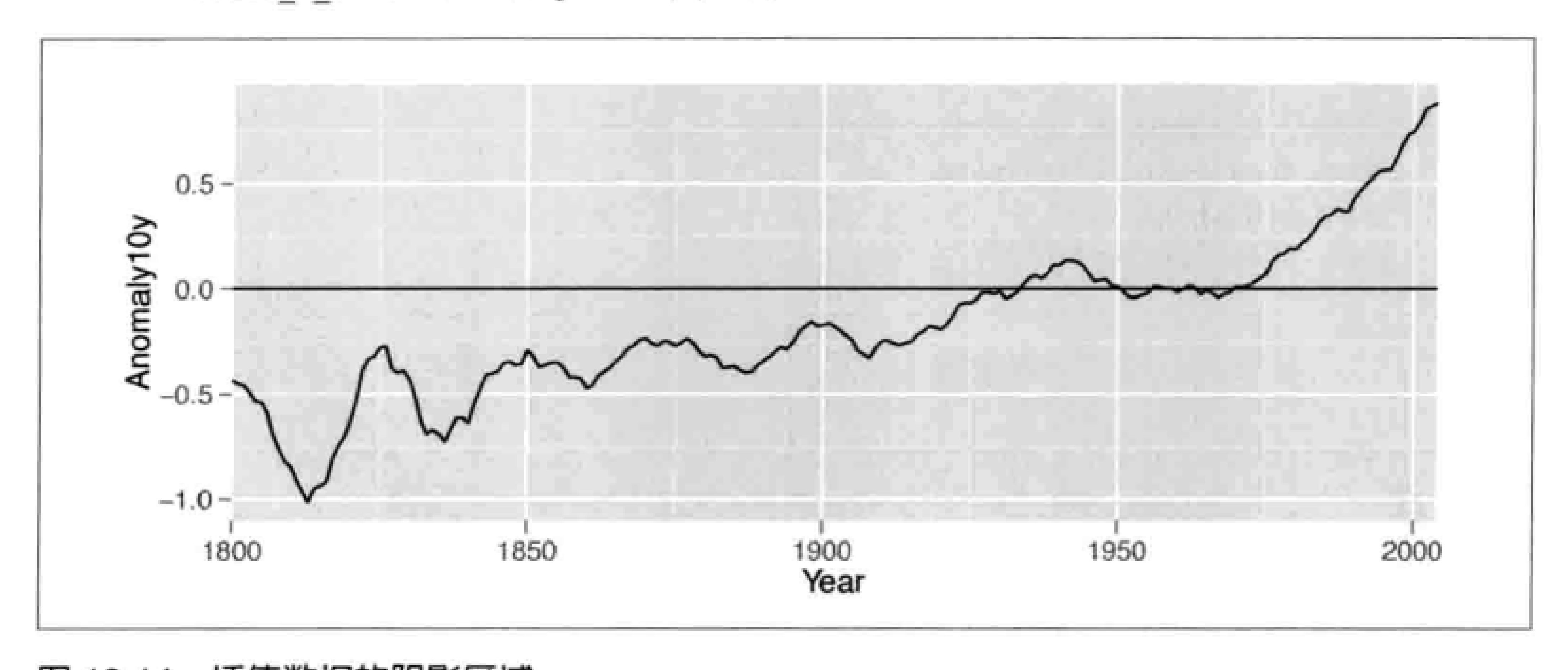


图 12-14　插值数据的阴影区域

第13章

其他图形

数据可视化的方法很多，有些图形实在难以清晰地分门别类。本章展示如何绘制这些图形。

13.1 绘制相关矩阵图

问题

如何绘制一个相关矩阵图？

方法

我们先来看看 mtcars 数据集：

```
mtcars

                    mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4          21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag      21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710         22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
...
Ferrari Dino       19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
Maserati Bora      15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
Volvo 142E         21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

首先，使用 cor 函数来计算相关矩阵，将会得到每两列之间的相关系数：

```
mcor <- cor(mtcars)

# 输出mcor，保留两位小数
round(mcor, digits=2)

      mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
mpg  1.00 -0.85 -0.85 -0.78  0.68 -0.87  0.42  0.66  0.60  0.48 -0.55
```

```
cyl  -0.85  1.00  0.90  0.83 -0.70  0.78 -0.59 -0.81 -0.52 -0.49  0.53
disp -0.85  0.90  1.00  0.79 -0.71  0.89 -0.43 -0.71 -0.59 -0.56  0.39
hp   -0.78  0.83  0.79  1.00 -0.45  0.66 -0.71 -0.72 -0.24 -0.13  0.75
drat  0.68 -0.70 -0.71 -0.45  1.00 -0.71  0.09  0.44  0.71  0.70 -0.09
wt   -0.87  0.78  0.89  0.66 -0.71  1.00 -0.17 -0.55 -0.69 -0.58  0.43
qsec  0.42 -0.59 -0.43 -0.71  0.09 -0.17  1.00  0.74 -0.23 -0.21 -0.66
vs    0.66 -0.81 -0.71 -0.72  0.44 -0.55  0.74  1.00  0.17  0.21 -0.57
am    0.60 -0.52 -0.59 -0.24  0.71 -0.69 -0.23  0.17  1.00  0.79  0.06
gear  0.48 -0.49 -0.56 -0.13  0.70 -0.58 -0.21  0.21  0.79  1.00  0.27
carb -0.55  0.53  0.39  0.75 -0.09  0.43 -0.66 -0.57  0.06  0.27  1.00
```

如果数据含有不能用来计算系数的任何列（比如一列姓名），应该先将这些列剔除。如果在原始数据中存在缺失值（NA），得到的相关矩阵中也会有缺失值。为了克服这个问题，你可以使用函数选项 use="complete.obs" 或者 use="pairwise.complete.obs"。

我们使用 corrplot 包来绘制相关矩阵图（见图 13-1），第一次使用时用 install.packages("corrplot") 命令来安装该包：

```
library(corrplot)

corrplot(mcor)
```

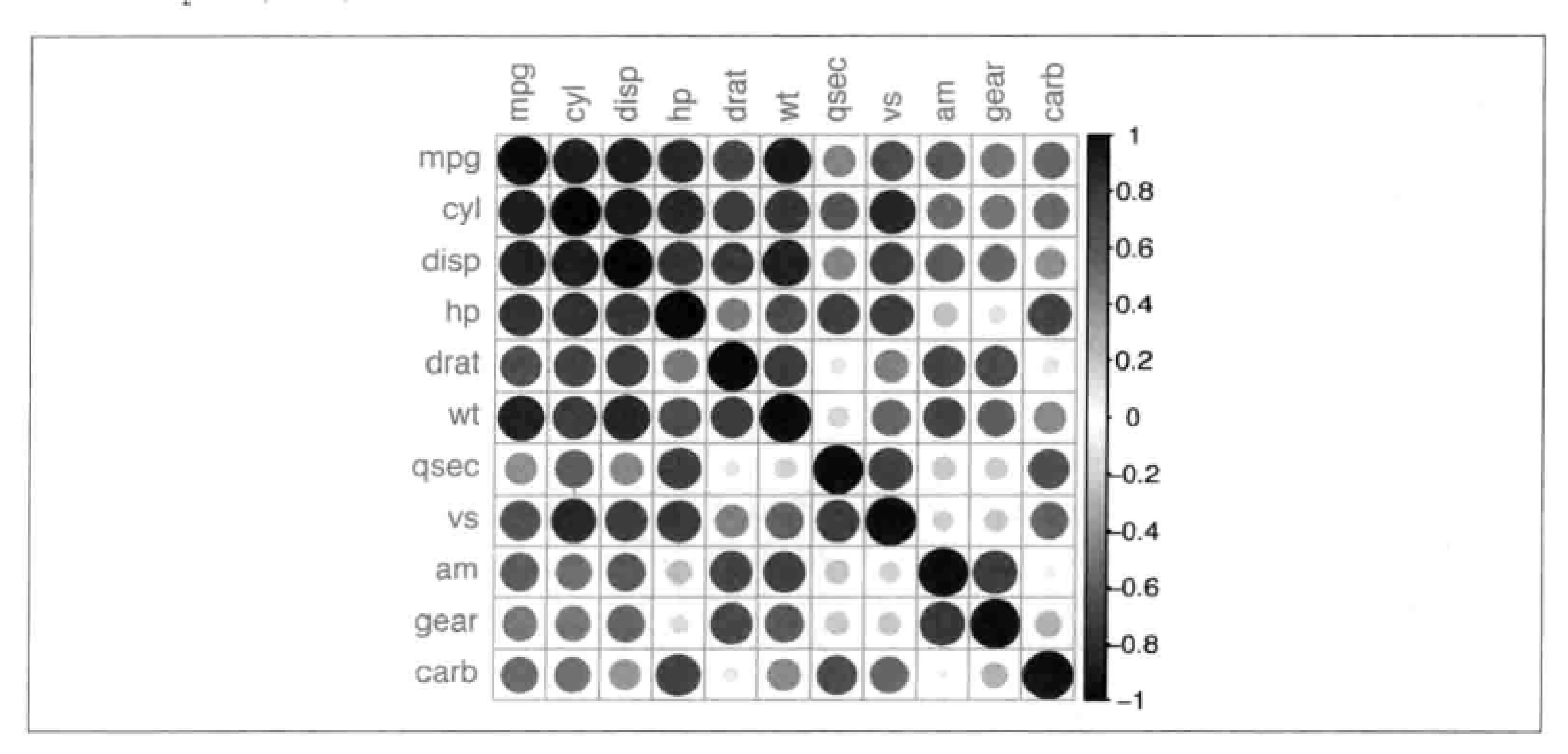


图 13-1　一个相关矩阵图

讨论

corrplot() 函数有相当多的选项。这里给出一个绘制相关矩阵图的例子，例图使用颜色方块和黑色文本标签，并且上边的文本标签呈 45° 右倾（见图 13-2）。[①]

```
corrplot(mcor, method="shade", shade.col=NA, tl.col="black", tl.srt=45)
```

将相关系数展示在矩阵的每一个小方块上也是很有用的。在本例中，我们使用一

① 原书代码冗余，此处对代码进行了调整。——译者注

个稍淡一点的调色板，这样会让系数更加可读；同时，我们也移除掉多余的颜色图例。此外，我们对矩阵重新排序，这样相近的变量在图中会更近，其中使用的参数是 order="AOE"（前两个特征向量的角排序）。结果如图 13-3 所示。

```
# 生成一个淡一点的调色板
col <- colorRampPalette(c("#BB4444", "#EE9988", "#FFFFFF", "#77AADD", "#4477AA"))

corrplot(mcor, method="shade", shade.col=NA, tl.col="black", tl.srt=45,
         col=col(200), addCoef.col="black", cl.pos "no", order="AOE")
```

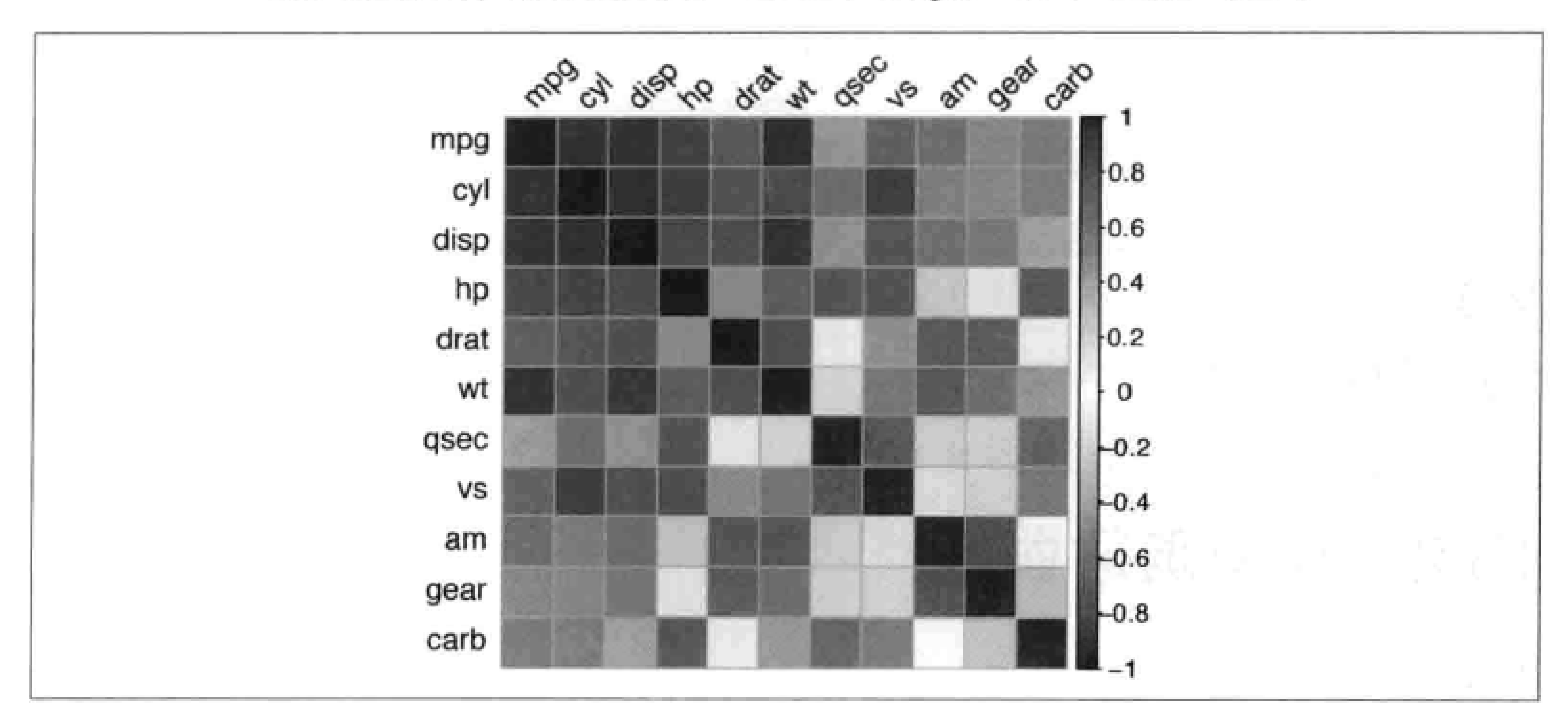


图 13-2　相关矩阵图：使用颜色方块、黑色倾斜的文本标签

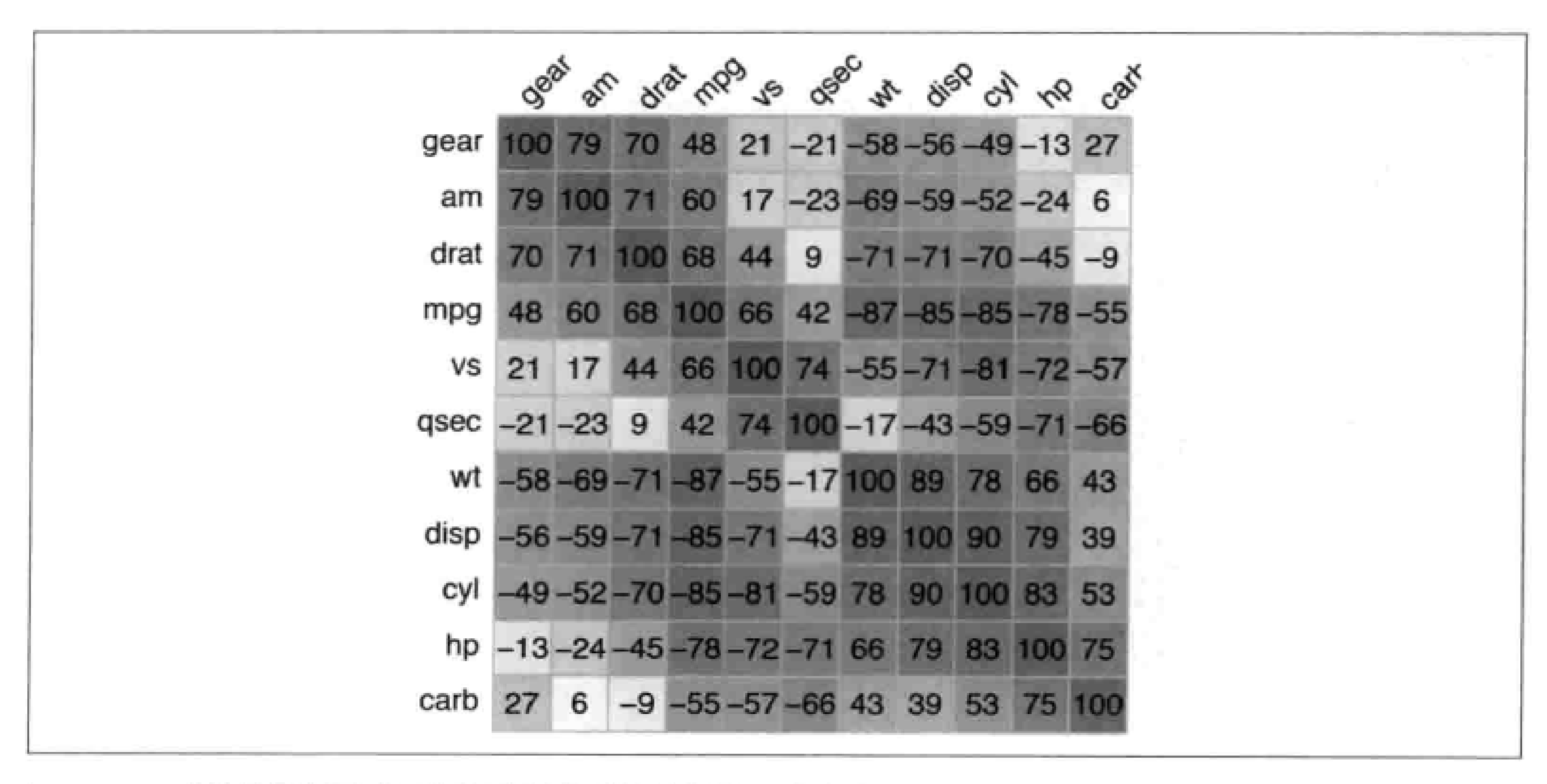


图 13-3　相关矩阵图（添加了相关系数并移除了图例）

和其他独立的绘图函数一样，corrplot() 有很多自己的选项，这里无法一一列出，表 13-1 给出了一些最有用的参数。

表 13-1　corrplot() 的选项

选　项	描　述
type={"lower"| "upper"}	仅使用下三角或上三角
diag=FALSE	是否展示对角线上的数值
method="shade"	使用颜色方块图形
method="ellipse"	使用椭圆图形
addCoef.col="color"	在图形上添加相关系数的颜色
tl.srt="number"	设定图形上方文本标签的倾斜角度
tl.col="color"	设定文本标签颜色
order={"AOE" | "FPC" | "hclust"}	矩阵重排序，使用特征值角排序、第一主成分或层次聚类

另见

如果要生成散点图矩阵，参见 5.13 节。

关于更多数据取子集的方法，参见 15.7 节。

13.2　绘制函数曲线

问题

如何绘制函数曲线？

方法

使用 stat_function() 函数。为了得到合适的 x 的范围，必须给 ggplot() 函数传递一个“哑”数据框。在本例中，我们用正态分布的密度函数 dnorm() 来演示（见图 13-4 左图）。

```
# 这个数据框仅仅用于设定范围
p <- ggplot(data.frame(x=c(-3,3)), aes(x=x))

p + stat_function(fun = dnorm)
```

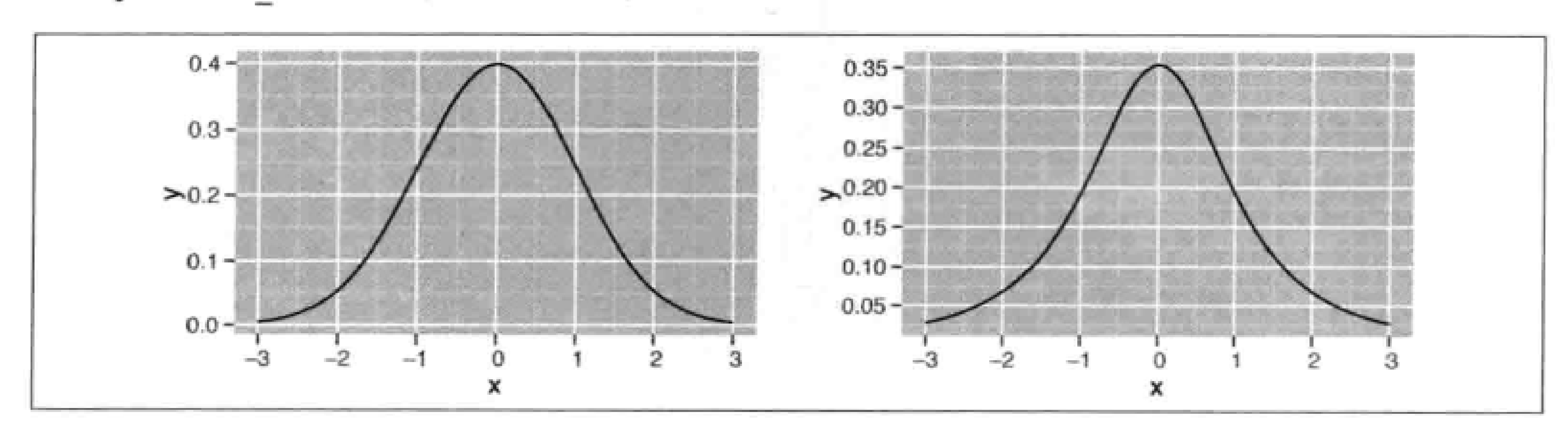


图 13-4　左图：正态分布函数　右图：自由度为 2 的 t 分布

讨论

某些函数需要设定额外的参数，比如 t 分布的密度函数 dt() 就需要一个参数来设定自

由度（见图 13-4 右图）。可以这样来设定额外的参数：先把它们放在一个列表（list）中，再传递给 args。

```
p + stat_function(fun=dt, args=list(df=2))
```

我们也可以绘制自定义的函数。其中第一个参数必须是 x 轴的值，并且必须返回 y 轴的值。在本例中，我们绘制一个 S 型函数（见图 13-5）。

```
myfun <- function(xvar) {
    1/(1 + exp(-xvar + 10))
}

ggplot(data.frame(x=c(0, 20)), aes(x=x)) + stat_function(fun=myfun)
```

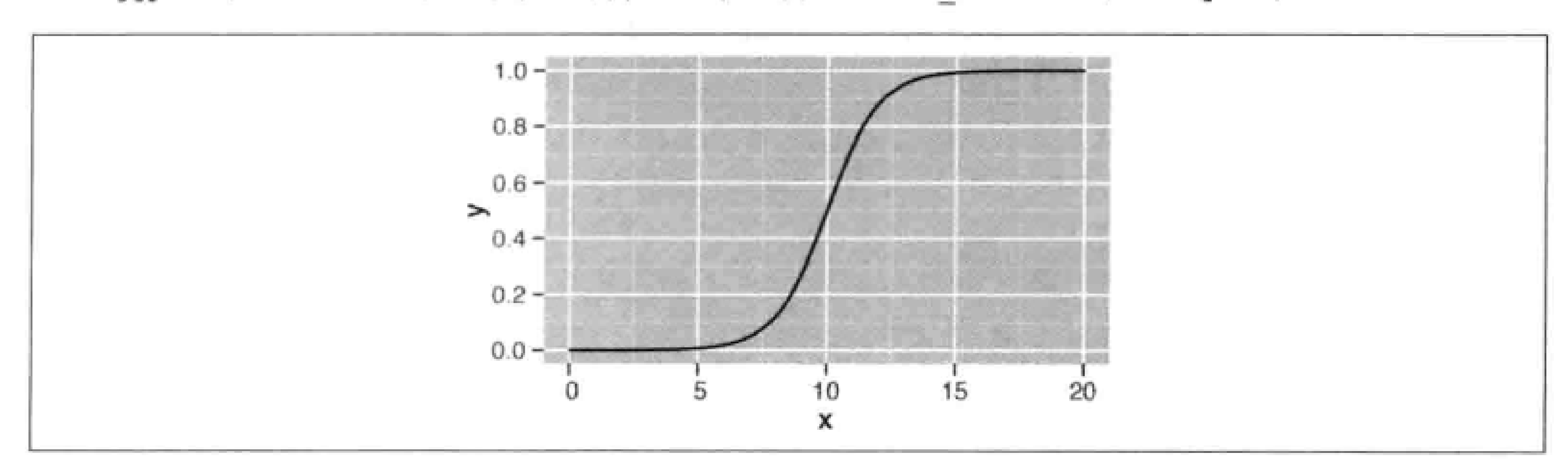


图 13-5 用户自定义的函数

计算函数值时默认使用给出的 x 范围内的 101 个点。但如果你的函数跳跃很大，画出来的图可能有一些支离破碎的线段。为了让曲线更加光滑，可以给 stat_function() 传递一个更大的 n，比如 stat_function(fun=myfun, n=200)。

另见

绘制模型对象的预测值（如 lm 和 glm），参见 5.7 节。

13.3 在函数曲线下添加阴影

问题

如何在函数曲线下的某一区域添加阴影？

方法

根据你的曲线函数定义一个新的函数，把 x 范围外对应的值替换为 NA，如图 13-6 所示。

```
# 在 0 < x < 2 时返回 dnorm(x)，其他时候返回 NA
dnorm_limit <- function(x) {
    y <- dnorm(x)
    y[x < 0  |  x > 2] <- NA
    return(y)
```

```
}

# ggplot() 使用“哑”数据
p <- ggplot(data.frame(x=c(-3, 3)), aes(x=x))

p + stat_function(fun=dnorm_limit, geom="area", fill="blue", alpha=0.2) +
    stat_function(fun=dnorm)
```

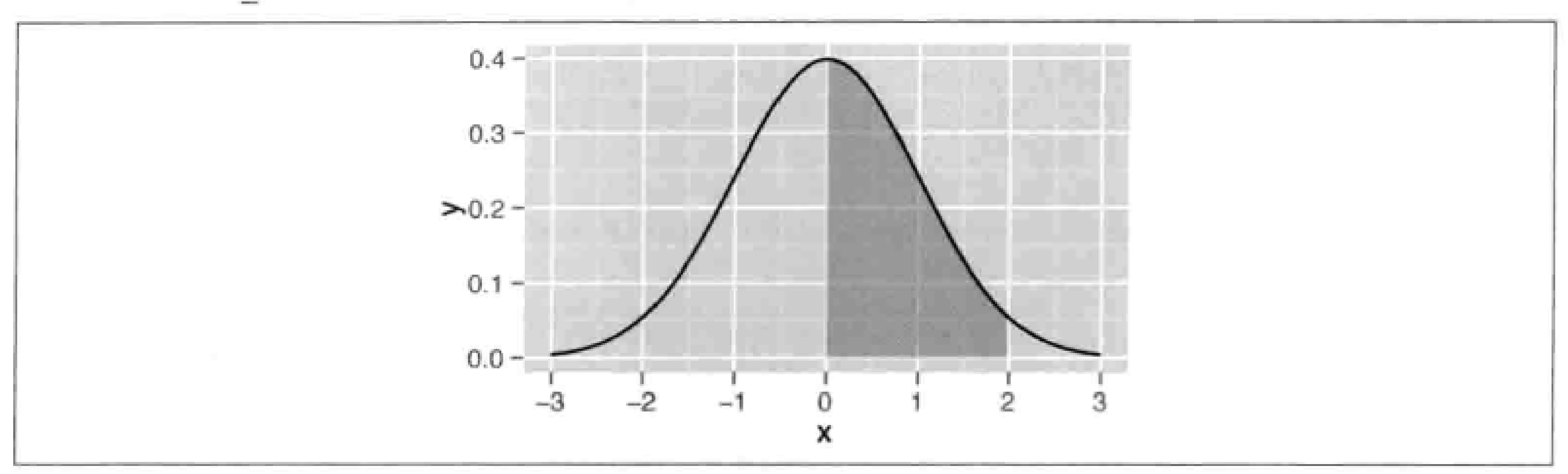


图 13-6　添加了阴影区域的函数曲线图

记住，给这个函数传入的是一个向量，并不是一个单独的值。如果这个函数每次只操作一个元素，那么使用一个 if/else 语句来根据 *x* 的值确定返回值是更为合理的。但是那样做在这里却行不通，因为 *x* 是一个含有许多值的向量。

讨论

R 中有第一类函数，我们可以写一个函数来返回一个闭包。也就是说，我们可以编写一个能够编写函数的函数。

这个函数将允许你传递一个函数、一个最小值和一个最大值。定义域外对应的值域返回 NA：

```
limitRange <- function(fun, min, max) {
    function(x) {
        y <- fun(x)
        y[x < min  |  x > max] <- NA
        return(y)
    }
}
```

现在我们可以调用这个函数来生成另一个函数了——这和之前使用的 dnorm_limit() 函数效果一样：

```
# 返回一个函数
dlimit <- limitRange(dnorm, 0, 2)

# 现在我们可以尝试新函数了 — 仅对 0-2 之间的输入返回输出值
dlimit(-2:4)

   [1]        NA        NA 0.39894228 0.24197072 0.05399097        NA        NA
```

我们使用 limitRange() 来生成函数，并传递给 stat_function()：

```
p + stat_function(fun = dnorm) +
    stat_function(fun = limitRange(dnorm, 0, 2),
                  geom="area", fill="blue", alpha=0.2)
```

limitRange() 函数可以用来生成任何函数的“区间限制式”函数，不局限于 dnorm()。之所以如此，是因为我们没有针对每一种情况来写不同的函数得到硬编码值，而是写了一个能够根据不同情况智能传递参数的函数。

如果你非常非常仔细地观察图 13-6，可能会看见阴影区域并没有和我们给出的边界范围完全对齐。这是因为 ggplot2 在固定区间上计算函数值时做了个数值近似，并且这些区间没有完全落在指定的范围上。如 13.2 节所述，我们可以通过设置 stat_function(n=200) 以增加内插点数量的方式来提升近似的效果。

13.4 绘制网络图

问题

如何绘制一个网络图？

方法

使用 igraph 包。要绘制网络图，首先给 graph() 函数传递一个包含所有边的向量，然后绘制结果对象（见图 13-7）：

```
# 首次运行可能需要安装包，命令：install.packages("igraph")
library(igraph)

# 指定一个有向图的边
gd <- graph(c(1,2, 2,3, 2,4, 1,4, 5,5, 3,6))
plot(gd)

# 一个无向图
gu <- graph(c(1,2, 2,3, 2,4, 1,4, 5,5, 3,6), directed=FALSE)
# 不画标签
plot(gu, vertex.label=NA)
```

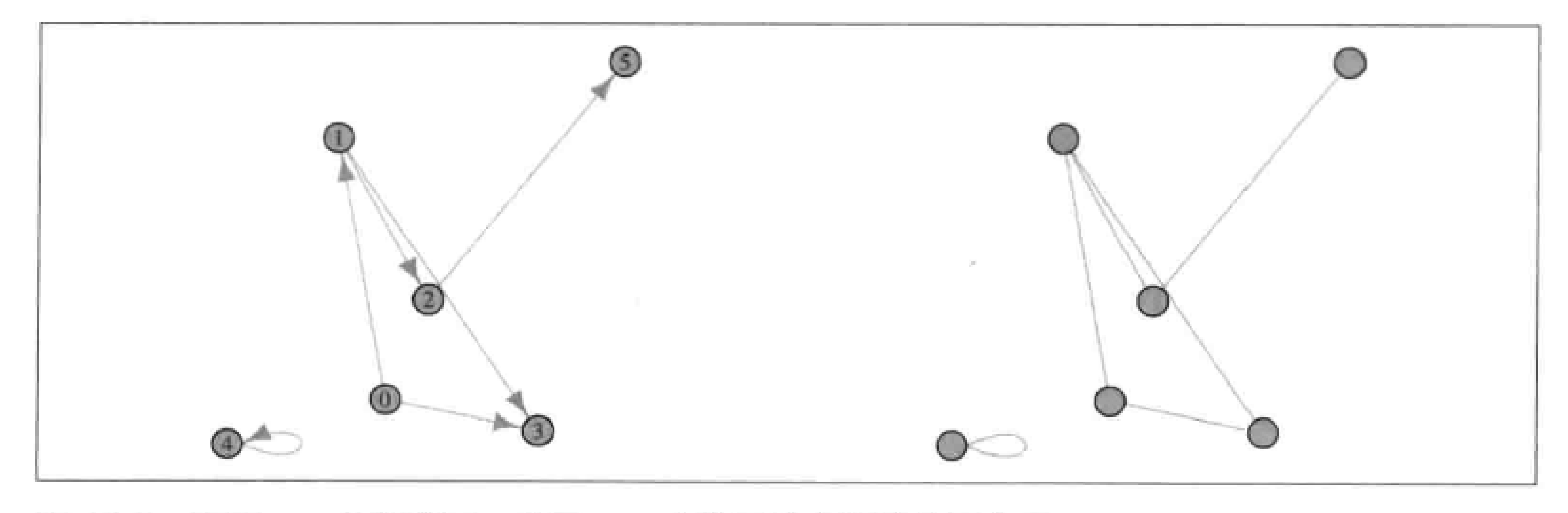


图 13-7　左图：一个有向图　右图：一个没画节点标签的无向图

这是两个图对象的结构：

```
str(gd)

IGRAPH D--- 6 6 --
+ edges:
[1] 1->2 2->3 2->4 1->4 5->5 3->6

str(gu)

IGRAPH U--- 6 6 --
+ edges:
[1] 1--2 2--3 2--4 1--4 5--5 3--6
```

讨论

在网络图中，节点的位置并不是由所给数据确定的，它们是随机放置的。如果要让得到的图更可读，可以在绘图之前设置随机数种子。比如，可以尝试不同的随机数种子直到得到一个比较满意的结果：

```
set.seed(229)
plot(gu)
```

也可以从数据框中直接生成图。使用数据框的前两列，并且每一列确定两个节点相连。在下一个例子中（见图 13-8），我们将使用这种结构的数据集 madmen2。我们还会使用 Fruchterman-Reingold 布局算法。该算法的主要思想是所有节点之间都有电磁斥力，但是连接节点的边像弹簧一样，会把相应的节点拉在一起：

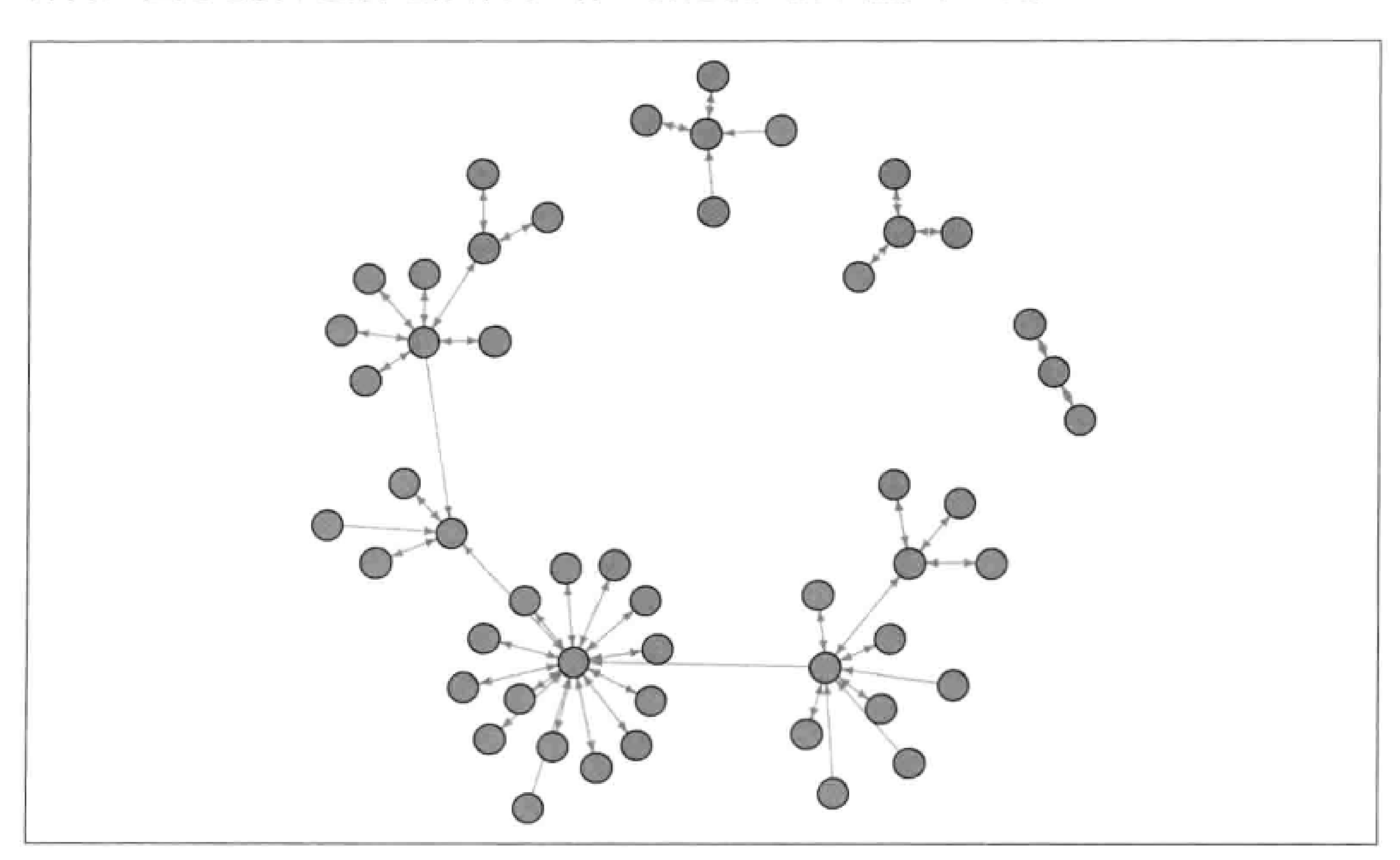


图 13-8　从数据框生成的有向图，布局算法为 Fruchterman-Reingold 算法

```
library(gcookbook)  # 为了使用数据集
madmen2

       Name1                  Name2
 Abe Drexler            Peggy Olson
     Allison             Don Draper
 Arthur Case           Betty Draper
 ...

# 从数据集中生成图对象
g <- graph.data.frame(madmen2, directed=TRUE)

# 移除多余的空白边
par(mar=c(0,0,0,0))

plot(g, layout=layout.fruchterman.reingold, vertex.size=8, edge.arrow.size=0.5,
     vertex.label=NA)
```

也可以从数据框生成无向图。madmen 数据集中一行就可以表示一条边，因为对于无向图来说方向是没有意义的。现在，我们使用圆圈布局（见图 13-9）：

```
g <- graph.data.frame(madmen, directed=FALSE)
par(mar=c(0,0,0,0))  # 移除多余的空白边
plot(g, layout=layout.circle, vertex.size=8, vertex.label=NA)
```

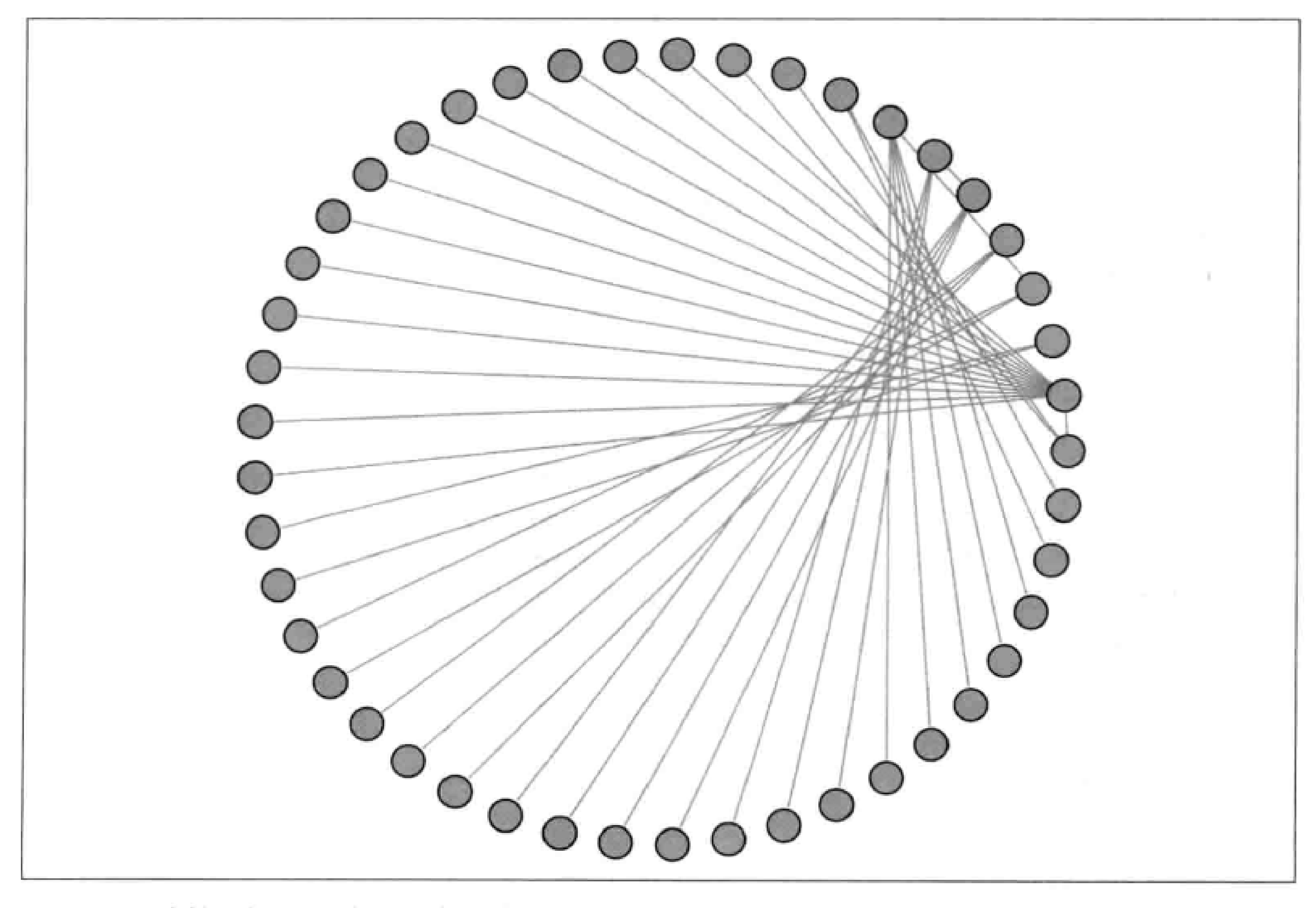


图 13-9　从数据框生成的圆形布局的无向图

另见

更多关于图像输出参数的信息，参见 ?plot.igraph。关于布局选项的参数，参见 ?igraph::layout。

igraph 的一个替代选择是 Rgraphviz，该包是 Graphviz（一个绘制网络图的开源库）的前端。它的优点是能更方便地处理标签，而且也更容易控制布局。但是它比较难安装。Rgraphviz 由 Bioconductor 系统库维护。

13.5 在网络图中使用文本标签

问题

如何在网络图中使用文本标签？

方法

边和节点可能都有名字，但默认时这些名字可能没有被当作标签。为了设置标签，可以给 vertex.label 参数传递一个命名向量（见图 13-10）：

```
library(igraph)
library(gcookbook) # 为了使用数据集
# 复制 madmen 并删除偶数行
m <- madmen[1:nrow(madmen) %% 2 == 1, ]
g <- graph.data.frame(m, directed=FALSE)

# 输出节点名称
V(g)$name

 [1] "Betty Draper"    "Don Draper"         "Harry Crane"       "Joan Holloway"
 [5] "Lane Pryce"      "Peggy Olson"        "Pete Campbell"     "Roger Sterling"
 [9] "Sal Romano"      "Henry Francis"      "Allison"           "Candace"
[13] "Faye Miller"     "Megan Calvet"       "Rachel Menken"     "Suzanne Farrell"
[17] "Hildy"           "Franklin"           "Rebecca Pryce"     "Abe Drexler"
[21] "Duck Phillips"   "Playtex bra model" "Ida Blankenship"   "Mirabelle Ames"
[25] "Vicky"           "Kitty Romano"

plot(g, layout=layout.fruchterman.reingold,
     vertex.size        = 4,          # 让节点更小
     vertex.label       = V(g)$name,  # 设置标签
     vertex.label.cex   = 0.8,        # 小号字体
     vertex.label.dist  = 0.4,        # 标签和节点的位置错开
     vertex.label.color = "black")
```

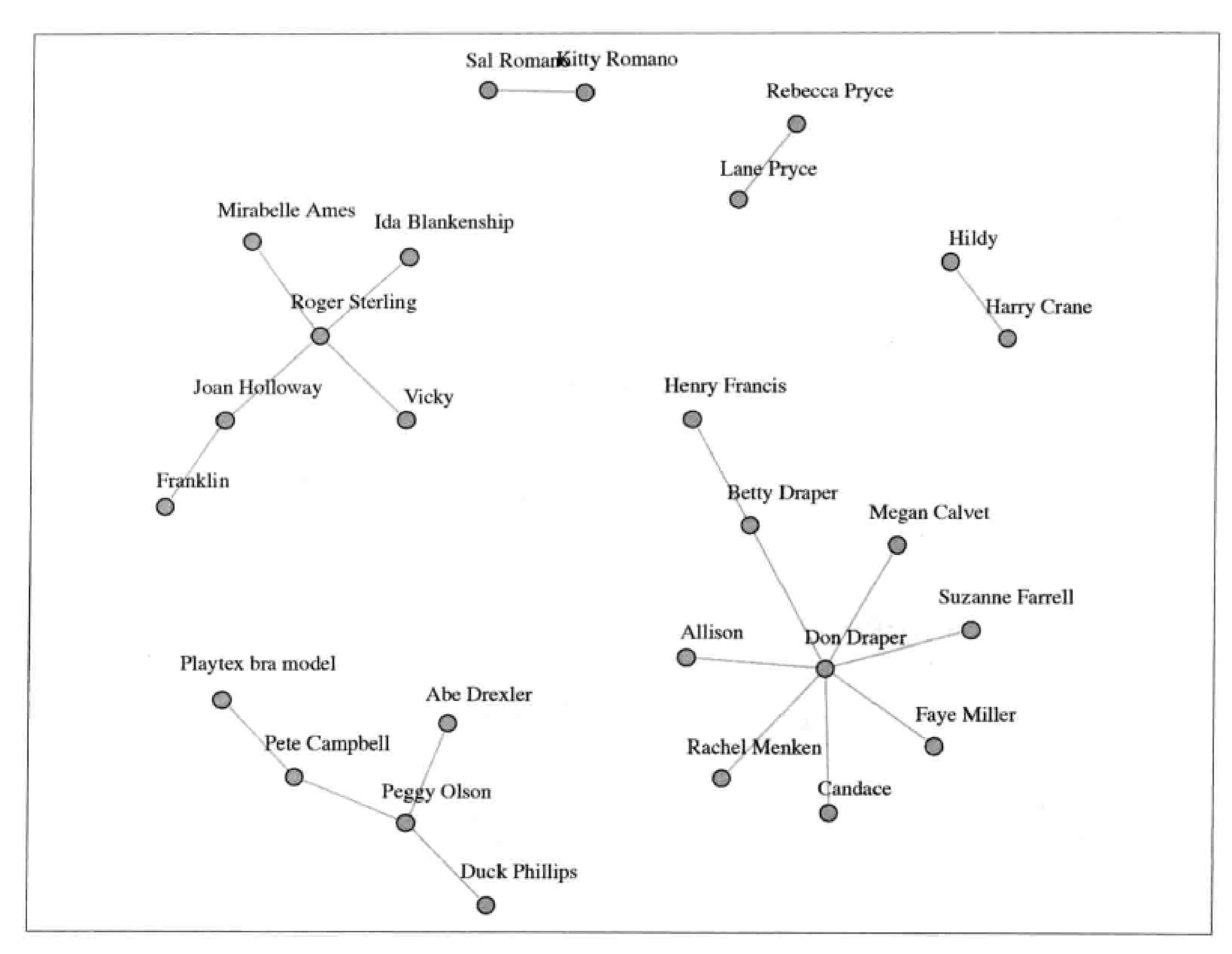


图 13-10　带有文本标签的网络图

讨论

另一种能得到相同效果的方法是修改绘图对象，这样就不用给 plot() 函数传递参数值了。此时，可以使用 V()$xxx <- 来代替 vertex.xxx 参数传递值。比如，下面的代码可以得到和之前代码相同的输出。

```
# 这和之前的代码是等价的
V(g)$size        <- 4
V(g)$label       <- V(g)$name
V(g)$label.cex   <- 0.8
V(g)$label.dist  <- 0.4
V(g)$label.color <- "black"

# 设置整个图的属性
g$layout <- layout.fruchterman.reingold

plot(g)
```

同样，也可以设置边的属性，使用 E() 函数或者给 edge.xxx 参数传递相应的值（见图 13-11）。

```
# 查看边
E(g)
```

```
# 将几个边的名字赋值为 "M"
E(g)[c(2,11,19)]$label <- "M"

# 将所有边颜色设置为灰色，然后把其中几个变为红色
E(g)$color             <- "grey70"
E(g)[c(2,11,19)]$color <- "red"

plot(g)
```

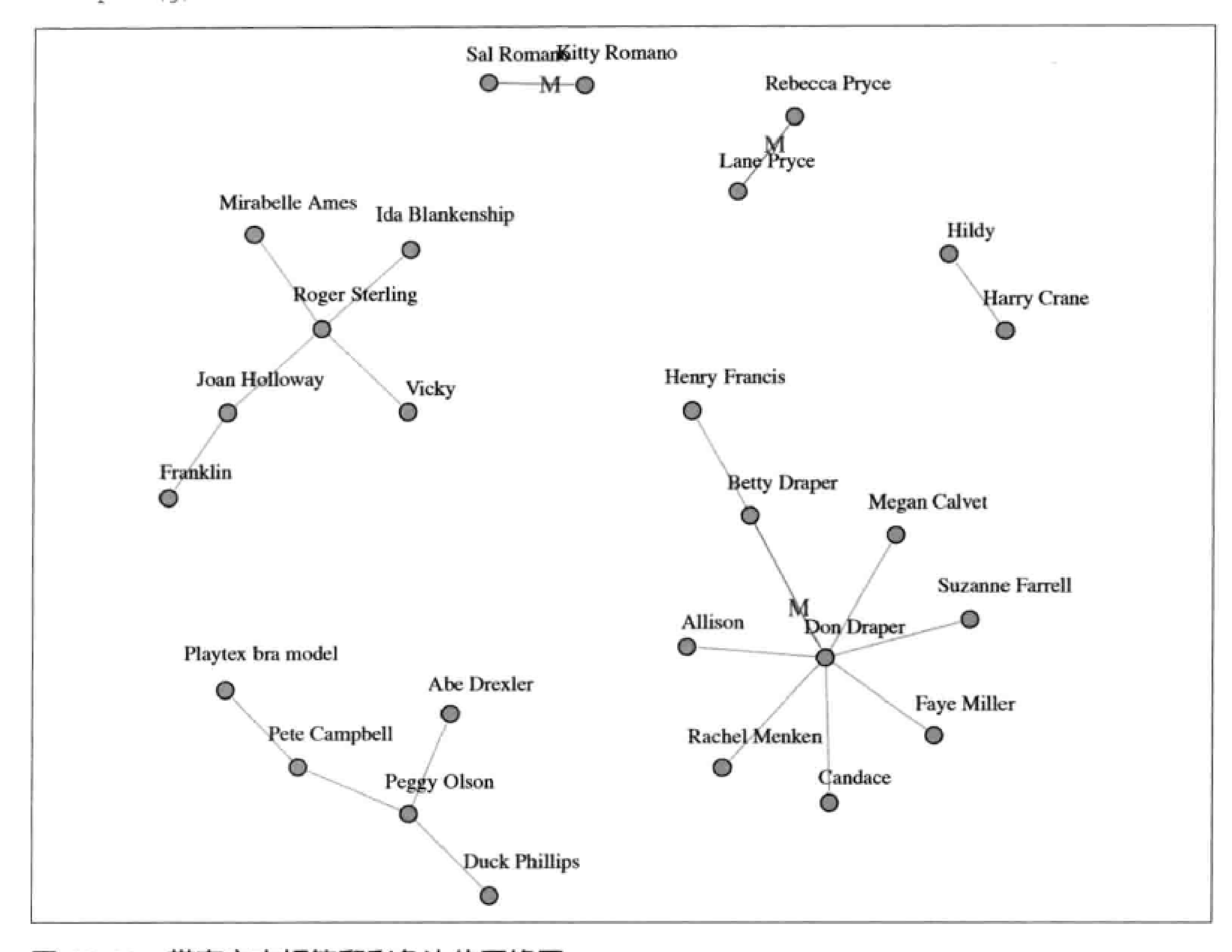


图 13-11　带有文本标签和彩色边的网络图

另见

查阅 ?igraph.plotting 来获取 igraph 包中更多的绘图参数。

13.6　如何绘制热图

问题

如何绘制热图？

方法

使用 geom_tile() 或者 geom_raster()，并将一个连续变量映射到 fill 上。我们将使用 presidents 数据集，它是一个时间序列对象而不是数据框。

```
Presidents

     Qtr1 Qtr2 Qtr3 Qtr4
1945   NA   87   82   75
1946   63   50   43   32
 ...
1973   68   44   40   27
1974   28   25   24   24

str(presidents)

Time-Series [1:120] from 1945 to 1975: NA 87 82 75 63 50 43 32 35 60 ...
```

我们首先将它转化为 ggplot() 可用的数据框格式，其中的列都是数值形式的。

```
pres_rating <- data.frame(
    rating  = as.numeric(presidents),
    year    = as.numeric(floor(time(presidents))),
    quarter = as.numeric(cycle(presidents))
)

pres_rating

 rating year quarter
     NA 1945       1
     87 1945       2
     82 1945       3
...
     25 1974       2
     24 1974       3
     24 1974       4
```

现在我们可以使用 geom_tile() 或 geom_raster() 来绘图（见图 13-12）。简单地将几个变量分别映射到 x, y 和 fill:

```
# 基础图形
p <- ggplot(pres_rating, aes(x=year, y=quarter, fill=rating))

# 使用 geom_tile()
p + geom_tile()

# 使用 geom_raster() 看起来一样，但效率略高
p + geom_raster()
```

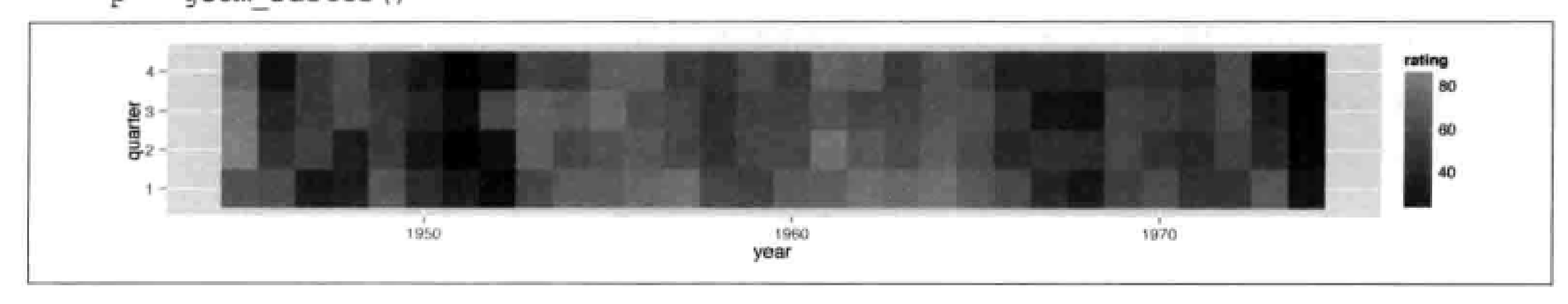


图 13-12　热图，数据中的缺失值用灰色来代替



geom_tile() 和 geom_raster() 的结果看起来一样，但实际上，它们是有区别的。阅读 6.12 节来获取更多相关主题的信息。

讨论

为了更有效地表达信息，你可能需要自定义热图的外观。在本例中，我们倒转 y 轴，这样顺序就是从上到下了，并且我们在 x 轴上每隔 4 年添加一个坐标刻度值来表示一个总统任期。此外，我们更换之前的颜色标度，使用 scale_fill_gradient2() 调色板，该调色板可以设置一个中点和两个端点的色彩值（见图 13-13）：

```
p + geom_tile() +
    scale_x_continuous(breaks = seq(1940, 1976, by = 4)) +
    scale_y_reverse() +
    scale_fill_gradient2(midpoint=50, mid="grey70", limits=c(0,100))
```

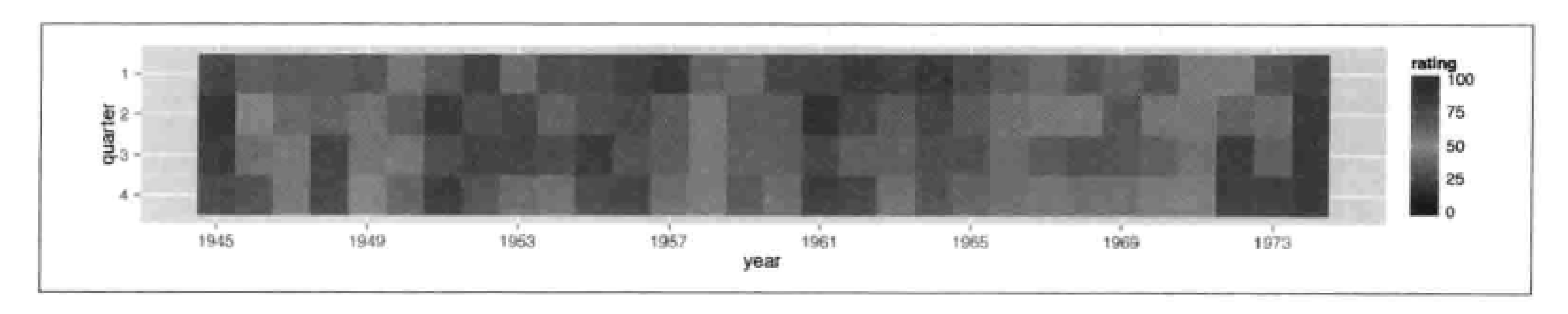


图 13-13　一个自定义外观的热图

另见

如果想使用不同的调色板，参见 12.6 节。

13.7　绘制三维散点图

问题

如何绘制一个三维散点图？

方法

我们使用 rgl 包，该包提供了 OpenGL 图形库的 3D 绘图接口。要画三维散点图（见图 13-14），可以使用 plot3d() 函数。其输入参数可以是两种形式：（1）一个数据框，前三列分别表示 x、y、z 的坐标；（2）直接传递三个向量，分别表示 x、y、z 的坐标。

```
# 首次运行可能需要安装包，命令：install.packages("rgl")
library(rgl)
plot3d(mtcars$wt, mtcars$disp, mtcars$mpg, type="s", size=0.75, lit=FALSE)
```

读图者可以通过点击和拖动鼠标来旋转图形，滑动鼠标滚轮来缩放图形。

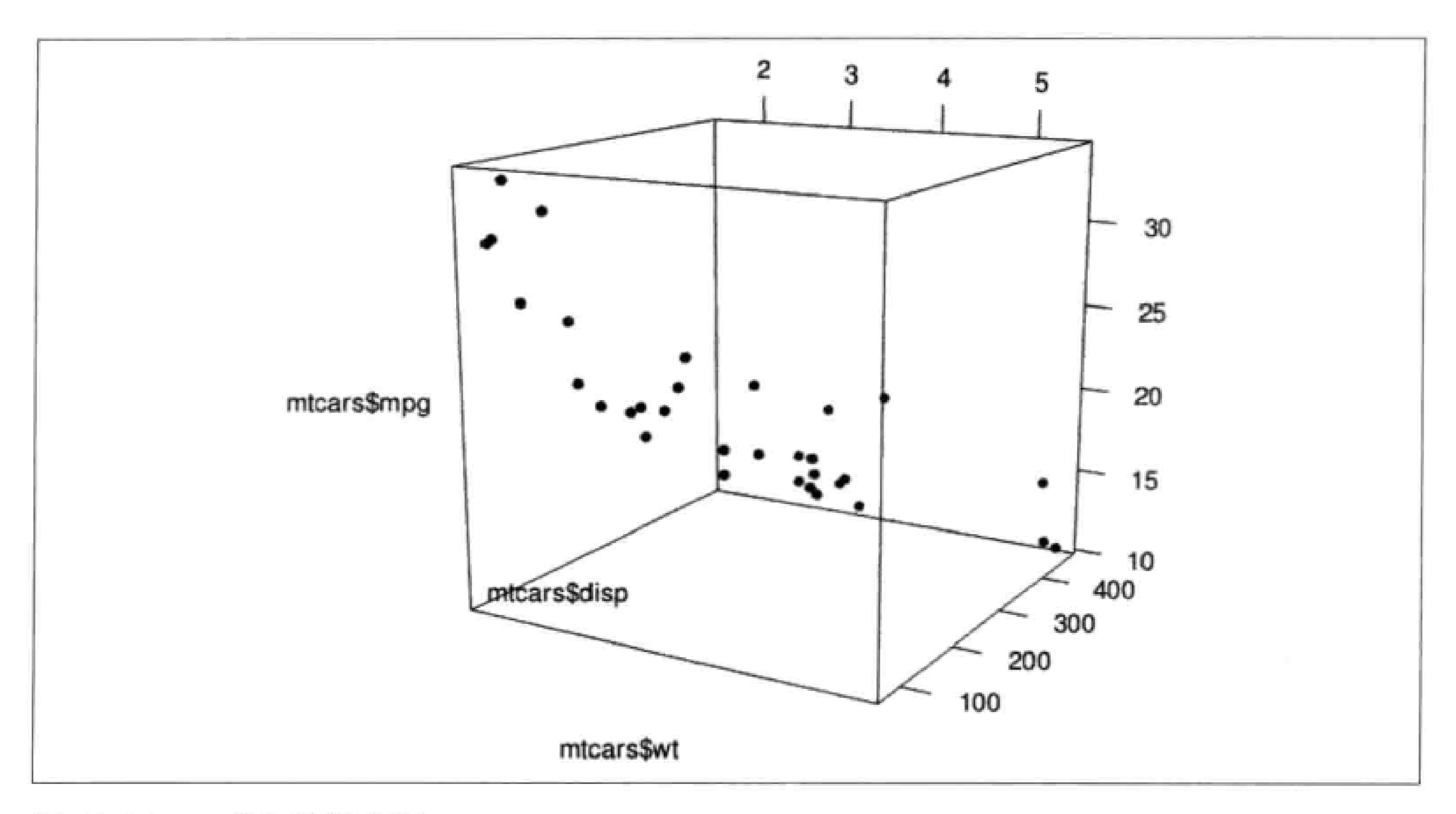
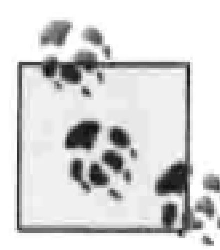


图 13-14　一个三维散点图



默认情况下，plot3d() 函数使用立方体式的点，这不大适合保存为 PDF。为了改进外观，我们使用 type="s" 来选择球形点，size=0.75 让点变小一些，lit=FALSE 来关闭 3D 灯光（否则点会闪闪发亮）。

讨论

三维散点图比较难解释，因此大多时候使用二维散点图可能会更好。这同样也意味着，还有很大改进空间可以让 3D 散点图更容易被理解。

在图 13-15 中，我们添加数值线段来增强空间点的位置表达力度：

```
# 交错出现两个向量的值
interleave <- function(v1, v2)  as.vector(rbind(v1,v2))

# 绘制点
plot3d(mtcars$wt, mtcars$disp, mtcars$mpg,
       xlab="Weight", ylab="Displacement", zlab="MPG",
       size=.75, type="s", lit=FALSE)

# 添加线段
segments3d(interleave(mtcars$wt,   mtcars$wt),
           interleave(mtcars$disp, mtcars$disp),
           interleave(mtcars$mpg,  min(mtcars$mpg)),
           alpha=0.4, col="blue")
```

可以微调图形的背景和坐标轴。在图 13-16 中，我们改变坐标轴刻度的数目、添加刻度值，并在指定的边添加坐标轴标签：

```
# 不画坐标刻度和标签
plot3d(mtcars$wt, mtcars$disp, mtcars$mpg,
       xlab = "", ylab = "", zlab = "",
       axes = FALSE,
       size=.75, type="s", lit=FALSE)

segments3d(interleave(mtcars$wt,   mtcars$wt),
           interleave(mtcars$disp, mtcars$disp),
           interleave(mtcars$mpg,  min(mtcars$mpg)),
           alpha = 0.4, col = "blue")

# 绘制盒子
rgl.bbox(color="grey50",          # 表面颜色为grey60，黑色文本
         emission="grey50",       # 光照颜色为grey50
         xlen=0, ylen=0, zlen=0)  # 不添加刻度

# 设置默认颜色为黑
rgl.material(color="black")

# 在指定边添加坐标轴标签。可能的值类似于 "x--", "x-+", "x+-" 和 "x++"
axes3d(edges=c("x--", "y+-", "z--"),
       ntick=6,                       # 每个轴上 6 个刻度线
       cex=.75)                       # 较小的字体

# 添加坐标标签。 'line' 指定标签和坐标轴的距离。
mtext3d("Weight",       edge="x--", line=2)
mtext3d("Displacement", edge="y+-", line=3)
mtext3d("MPG",          edge="z--", line=3)
```

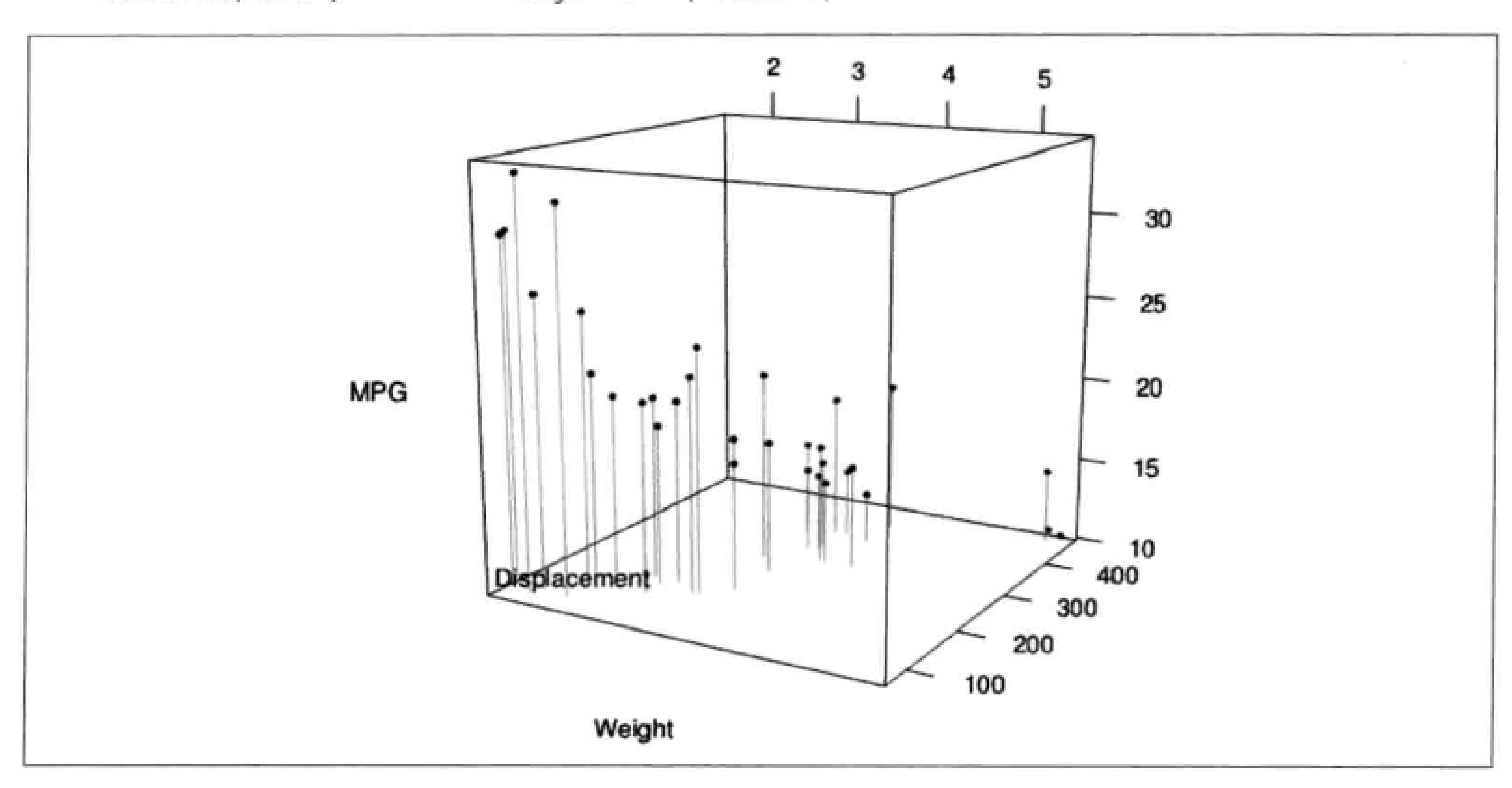


图 13-15　一个给每个点添加了竖直线的 3D 散点图

另见

查阅 ?plot3d 获取控制图形输出的更多选项。

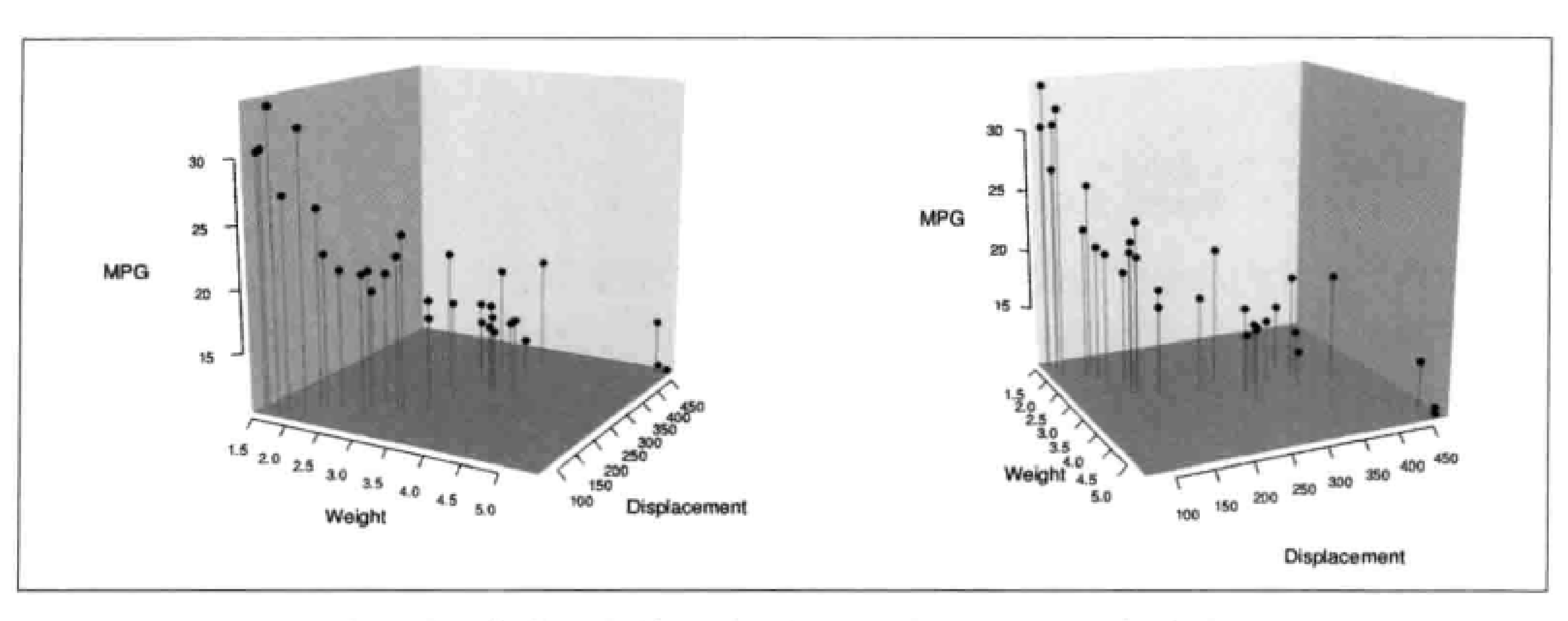


图 13-16　左图：坐标刻度和标签重新放置的三维图　右图：另一个角度的视图

13.8　在三维图上添加预测曲面

问题

如何在一个三维散点图上添加一个预测曲面？

方法

首先，我们需要定义一些功能函数来得到模型的预测值：

```
# 给定一个模型，根据 xvar 和 yvar 预测 zvar
# 默认为变量 x 和 y 的范围，生成 16x16 的网格
predictgrid <- function(model, xvar, yvar, zvar, res = 16, type = NULL) {
  # 计算预测变量的范围。下面的代码对 lm、glm 以及其他模型方法都适用，
  # 但针对其他模型方法时可能需要适当调整。
  xrange <- range(model$model[[xvar]])
  yrange <- range(model$model[[yvar]])

  newdata <- expand.grid(x = seq(xrange[1], xrange[2], length.out = res),
                         y = seq(yrange[1], yrange[2], length.out = res))
  names(newdata) <- c(xvar, yvar)
  newdata[[zvar]] <- predict(model, newdata = newdata, type = type)
  newdata
}

# 将长数据框中的 x, y, z 转化为列表，
# 其中 x, y 为行列值，z 为矩阵
df2mat <- function(p, xvar = NULL, yvar = NULL, zvar = NULL) {
  if (is.null(xvar)) xvar <- names(p)[1]
  if (is.null(yvar)) yvar <- names(p)[2]
  if (is.null(zvar)) zvar <- names(p)[3]

  x <- unique(p[[xvar]])
  y <- unique(p[[yvar]])
  z <- matrix(p[[zvar]], nrow = length(y), ncol = length(x))

  m <- list(x, y, z)

  names(m) <- c(xvar, yvar, zvar)
```

```
  m
}

# 交错出现两个向量的元素
interleave <- function(v1, v2)  as.vector(rbind(v1,v2))
```

利用这些定义好的功能性函数，我们可以对数据生成线性模型，然后利用 surface3d() 函数在原散点图上添加网格式的预测图，如图 13-17 所示：

```
library(rgl)

# 复制数据集
m <- mtcars

# 生成线性模型
mod <- lm(mpg ~ wt + disp + wt:disp, data = m)

# 根据 wt 和 disp，得到 mpg 的预测值
m$pred_mpg <- predict(mod)

# 根据 wt 和 disp 的网格，得到 mpg 的预测值
mpgrid_df <- predictgrid(mod, "wt", "disp", "mpg")
mpgrid_list <- df2mat(mpgrid_df)

# 根据数据点绘图
plot3d(m$wt, m$disp, m$mpg, type="s", size=0.5, lit=FALSE)

# 添加预测点（较小）
spheres3d(m$wt, m$disp, m$pred_mpg, alpha=0.4, type="s", size=0.5, lit=FALSE)

# 添加表示误差的线段
segments3d(interleave(m$wt,   m$wt),
           interleave(m$disp, m$disp),
           interleave(m$mpg,  m$pred_mpg),
           alpha=0.4, col="red")

# 添加预测点网格
surface3d(mpgrid_list$wt, mpgrid_list$disp, mpgrid_list$mpg,
          alpha=0.4, front="lines", back="lines")
```

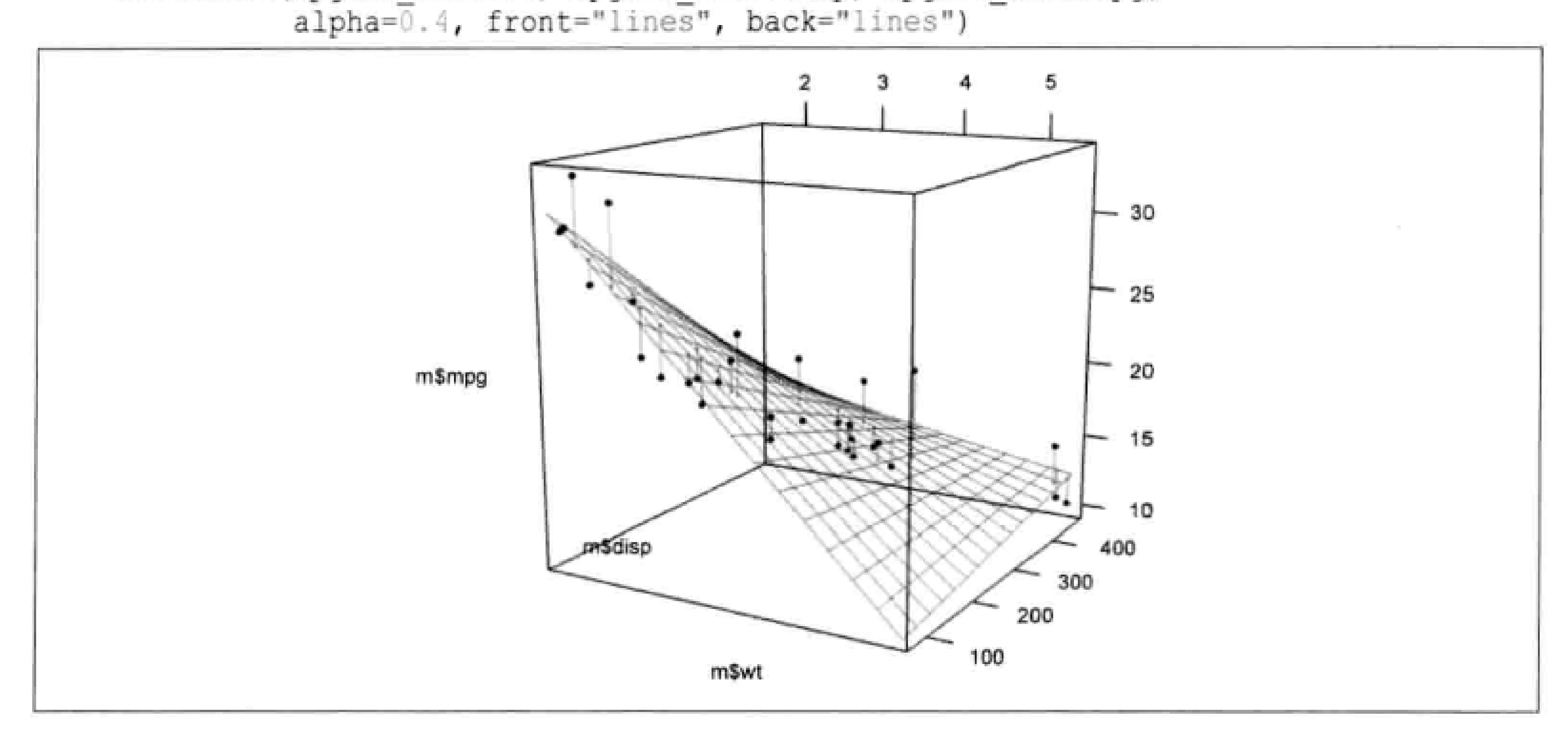


图 13-17　添加了预测曲面的三维图

讨论

我们可以调节图形的外观，如图 13-18 所示。我们还可以逐一添加图形的各个组件：

```
plot3d(mtcars$wt, mtcars$disp, mtcars$mpg,
       xlab = "", ylab = "", zlab = "",
       axes = FALSE,
       size=.5, type="s", lit=FALSE)

# 添加预测点（较小）
spheres3d(m$wt, m$disp, m$pred_mpg, alpha=0.4, type="s", size=0.5, lit=FALSE)

# 添加误差线段
segments3d(interleave(m$wt,   m$wt),
           interleave(m$disp, m$disp),
           interleave(m$mpg,  m$pred_mpg),
           alpha=0.4, col="red")

# 添加预测值网格
surface3d(mpgrid_list$wt, mpgrid_list$disp, mpgrid_list$mpg,
          alpha=0.4, front="lines", back="lines")

# 绘制盒子
rgl.bbox(color="grey50",              # 表面颜色为 grey60，黑色文本
         emission="grey50",           # 光照颜色为 grey50
         xlen=0, ylen=0, zlen=0)      # 不画刻度线

# 对象默认色设置为黑色
rgl.material(color="black")

# 在指定边添加坐标轴标签。可能的值类似于 "x--", "x-+", "x+-" 和 "x++"
axes3d(edges=c("x--", "y+-", "z--"),
       ntick=6,                        #  每个轴上 6 个刻度线
       cex=.75)                        #  较小字体

# 添加坐标标签。'line' 指定标签和坐标轴的距离。
mtext3d("Weight",       edge="x--", line=2)
mtext3d("Displacement", edge="y+-", line=3)
mtext3d("MPG",          edge="z--", line=3)
```

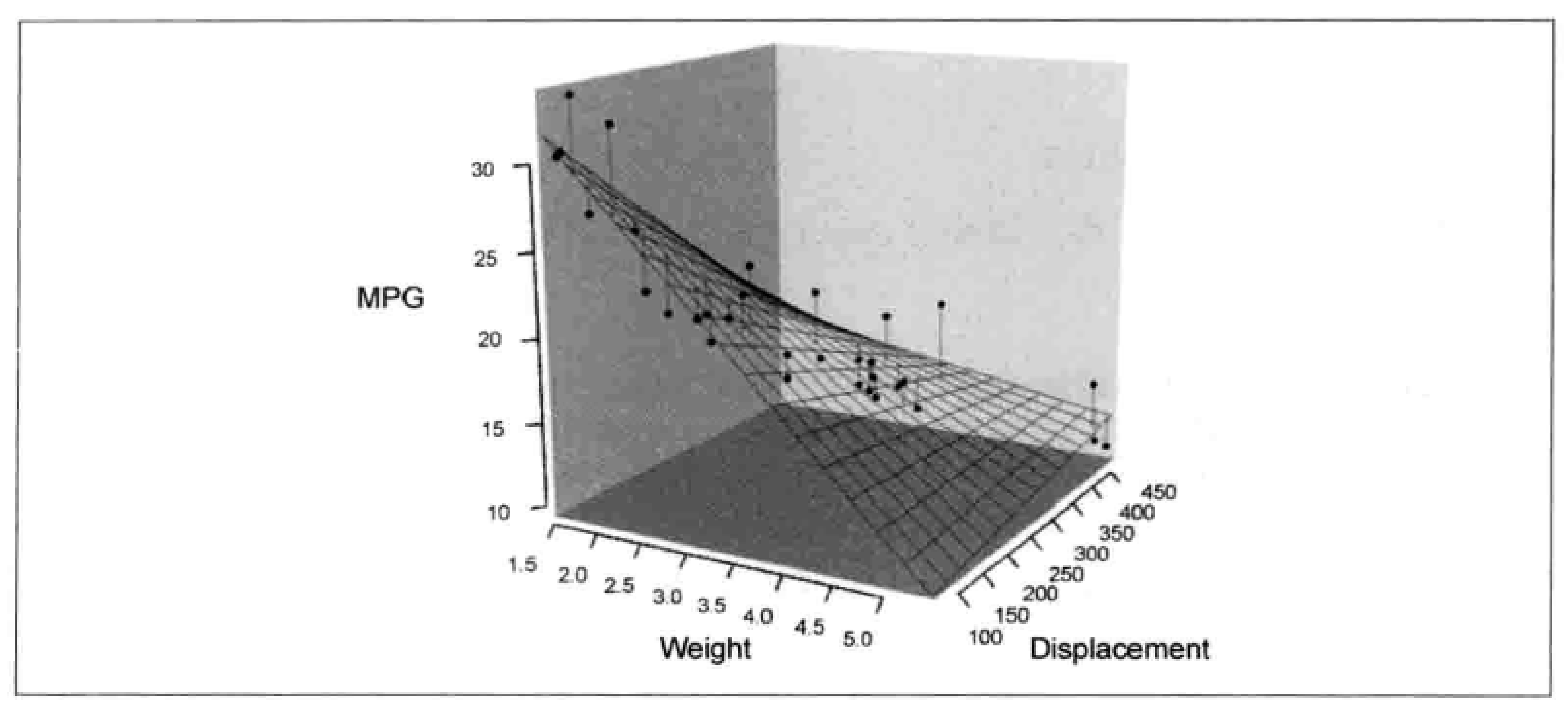


图 13-18　自定义外观的三维散点图

另见

输入 ?rgl.material 获取更多关于改变曲面图外观的信息。

13.9 保存三维图

问题

如何保存一个 rgl 包绘制的三维图？

方法

可以使用 rgl.snapshot() 来保存 rgl 包绘制的位图。它会精确捕捉屏幕上的图形。

```
library(rgl)
plot3d(mtcars$wt, mtcars$disp, mtcars$mpg, type="s", size=0.75, lit=FALSE)

rgl.snapshot('3dplot.png', fmt='png')
```

也可以使用 rgl.postscript() 保存为 PostScript 或 PDF 格式文件：

```
rgl.postscript('figs/miscgraph/3dplot.pdf', fmt='pdf')

rgl.postscript('figs/miscgraph/3dplot.ps', fmt='ps')
```

PostScript 和 PDF 输出文件并不支持 rgl 依赖的 OpenGL 库的很多特性。比如，它们不支持透明度，并且点线等对象的大小可能和屏幕上表现出来的也不一致。

讨论

为了使得输出的图片更加可读，你可以保存当前的视角，之后再恢复。

```
# 保存当前视角
view <- par3d("userMatrix")

# 恢复保存的视角
par3d(userMatrix = view)
```

为了将视角保存为代码，你可以使用 dput() 函数，然后将输出复制粘贴到你的代码中：

```
dput(view)

structure(c(0.907931625843048, 0.267511069774628, -0.322642296552658,
0, -0.410978674888611, 0.417272746562958, -0.810543060302734,
0, -0.0821993798017502, 0.868516683578491, 0.488796472549438,
0, 0, 0, 0, 1), .Dim = c(4L, 4L))
```

一旦有了 userMatrix 的文本表达式，将下面的代码添加到你的脚本中即可：

```
view <- structure(c(0.907931625843048, 0.267511069774628, -0.322642296552658,
0, -0.410978674888611, 0.417272746562958, -0.810543060302734,
0, -0.0821993798017502, 0.868516683578491, 0.488796472549438,
```

```
0, 0, 0, 0, 1), .Dim = c(4L, 4L))

par3d(userMatrix = view)
```

13.10 三维图动画

问题

如何让一个三维图根据视角的移动形成动画？

方法

旋转三维图能够更完整、多方位地观察数据。可以在 play3d() 中使用 spin3d() 来生成三维动画：

```
library(rgl)
plot3d(mtcars$wt, mtcars$disp, mtcars$mpg, type="s", size=0.75, lit=FALSE)

play3d(spin3d())
```

讨论

默认地，图像会绕着 *z* 轴（竖直的轴）旋转，直到你给 R 发送一个停止的指令。

你可以改变转轴、转速和持续时间：

```
# 绕 x 轴转动，每分钟 4 转，持续 20 秒钟
play3d(spin3d(axis=c(1,0,0), rpm=4), duration=20)
```

可以使用 movie3d() 来保存动画，方法和 play3d() 一样。它将会生成一系列 .png 格式的图片文件，每个文件代表一帧，然后可以利用 ImageMagick 软件提供的 convert 命令将这些文件合并转化为 .gif 动画文件。

这将会生成持续 15 秒，每秒 50 帧的动画：

```
# 绕 z 轴转动，每分钟 4 转，持续 15 秒
movie3d(spin3d(axis=c(0,0,1), rpm=4), duration=15, fps=50)
```

输出的文件将会被存放在一个临时文件夹中，地址名会在 R 窗口中输出。

如果你不想利用 ImageMagick 将输出的图片转换为 .gif，可以设置 convert=FALSE，然后用其他软件将这一系列的 .png 文件转换为动画。

13.11 绘制谱系图

问题

如何绘制一个谱系图（Dendrogram）来观察数据是如何层次聚类的？

方法

使用 `hclust()` 并画出它的结果。这可能需要一些数据预处理工作，在本例中，我们首先要从 `countries` 数据集中提取年份为 2009 年的子集，为简单起见，还要把包含 NA 的行全部删除，然后随机选择 25 行。

```
library(gcookbook) # 为了使用数据集

# 得到 2009 年的数据
c2 <- subset(countries, Year==2009)

# 去掉含有 NA 的行
c2 <- c2[complete.cases(c2), ]

# 随机选择 25 个国家
#（设定随机种子保证可重复性）
set.seed(201)
c2 <- c2[sample(1:nrow(c2), 25), ]

c2
```

```
                 Name Code Year        GDP laborrate  healthexp infmortality
6731         Mongolia  MNG 2009  1690.4170      72.9   74.19826         27.8
1733           Canada  CAN 2009 39599.0418      67.8 4379.76084          5.2
 ...
5966  Macedonia, FYR  MKD 2009  4510.2380      54.0  313.68971         10.6
10148    Turkmenistan  TKM 2009  3710.4536      68.0   77.06955         48.0
```

注意到行名（第一列）实质上是随机数，这是因为行是随机选择出来的。在画谱系图之前我们还需要再做一些事情。首先，我们需要设定行的名字——我们有叫做 Name 的列，但行名却是一些随机数（我们不是经常使用行名，但是对于 `hclust()` 函数，行名是必不可少的）。接下来，还要去掉在聚类中不需要的列，这些列是 Name、Code 和 Year。

```
rownames(c2) <- c2$Name
c2 <- c2[,4:7]
c2
```

```
                      GDP laborrate  healthexp infmortality
Mongolia        1690.4170      72.9   74.19826         27.8
Canada         39599.0418      67.8 4379.76084          5.2
  ...
Macedonia, FYR  4510.2380      54.0  313.68971         10.6
Turkmenistan    3710.4536      68.0   77.06955         48.0
```

GDP 的取值比 `infmortality` 大几个数量级，由于这个原因，`infmortality` 在聚类中的作用相对于 GDP 来说就会变得可以忽略不计，这可能不是我们想要的。为了解决这个问题，要对数据进行标准化：

```
c3 <- scale(c2)
c3
```

```
                   GDP   laborrate      healthexp infmortality
Mongolia        -0.6783472  1.15028714 -0.6341393599 -0.08334689
Canada           1.7504703  0.59747293  1.9736219974 -0.88014885
 ...
Macedonia, FYR  -0.4976803 -0.89837729 -0.4890859471 -0.68976254
Turkmenistan    -0.5489228  0.61915192 -0.6324002997  0.62883892
attr(,"scaled:center")
         GDP   laborrate     healthexp infmortality
   12277.960      62.288      1121.198       30.164
attr(,"scaled:scale")
         GDP   laborrate     healthexp infmortality
15607.852864    9.225523   1651.056974    28.363384
```

scale() 函数默认是将每一列相对于其标准差进行标准化，但也可以使用其他方法。

至此，我们已经做好了画谱系图的准备，结果如图 13-19 所示：

```
hc <- hclust(dist(c3))

# 画树状图
plot(hc)

# 对齐文本
plot(hc, hang = -1)
```

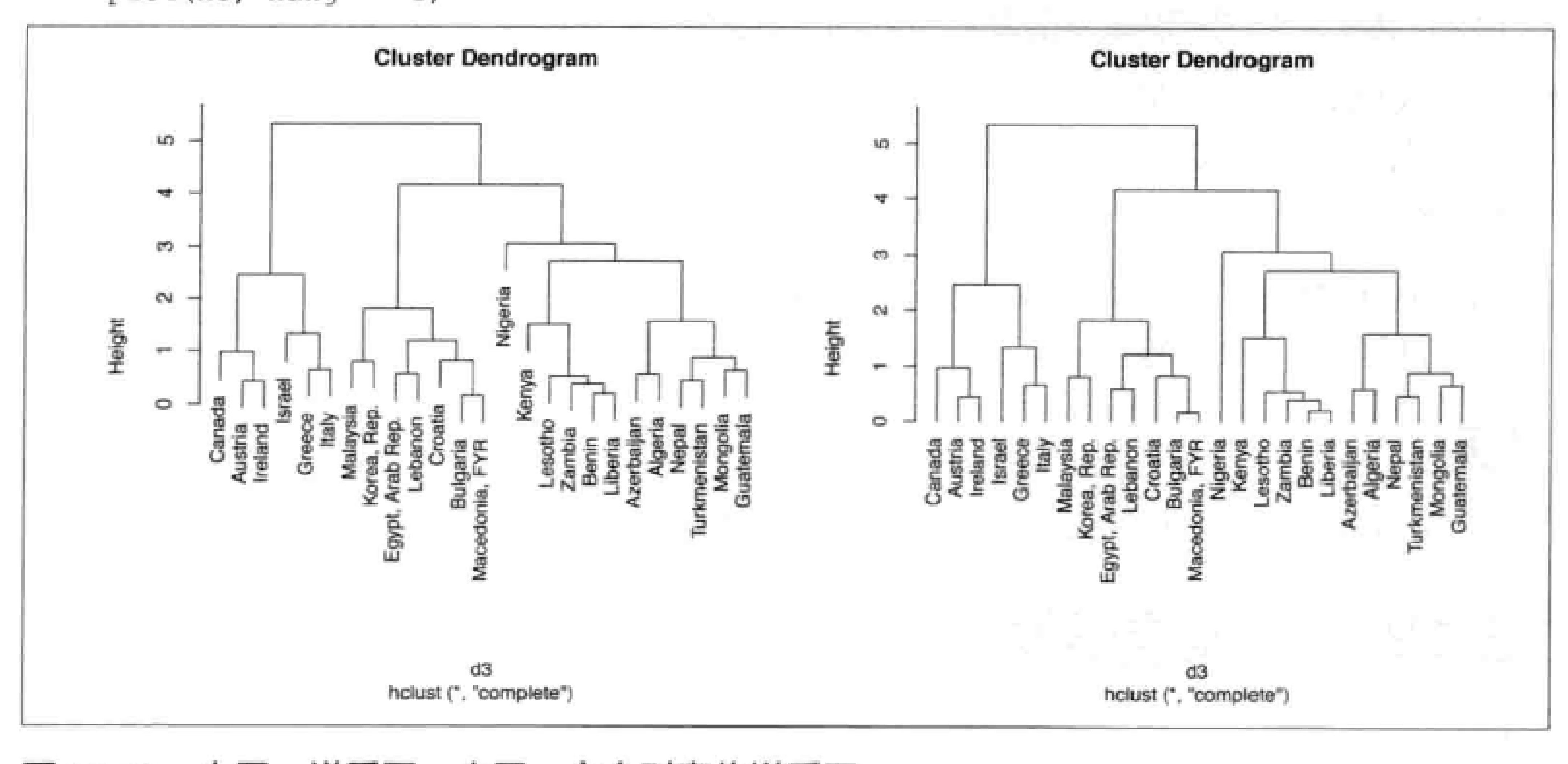


图 13-19　左图：谱系图　右图：文本对齐的谱系图

讨论

聚类分析是在 *n* 维空间中把点分配到类的一种简单方法（在这个例子中是四维）。层次聚类分析则将每组分成两个更小的组，在这里可以用谱系展示。在层次聚类的过程中有很多可以控制的参数，可能有不止一种“正确”的方法可以用来分析你的数据。

首先，我们用 scale() 的默认设置标准化了数据。你可以用不同的方法标准化数据，或者不标准化（在这个数据集里面，不标准化数据会导致 GDP 相对于其他变量占主导

地位，如图 13-20 所示）。

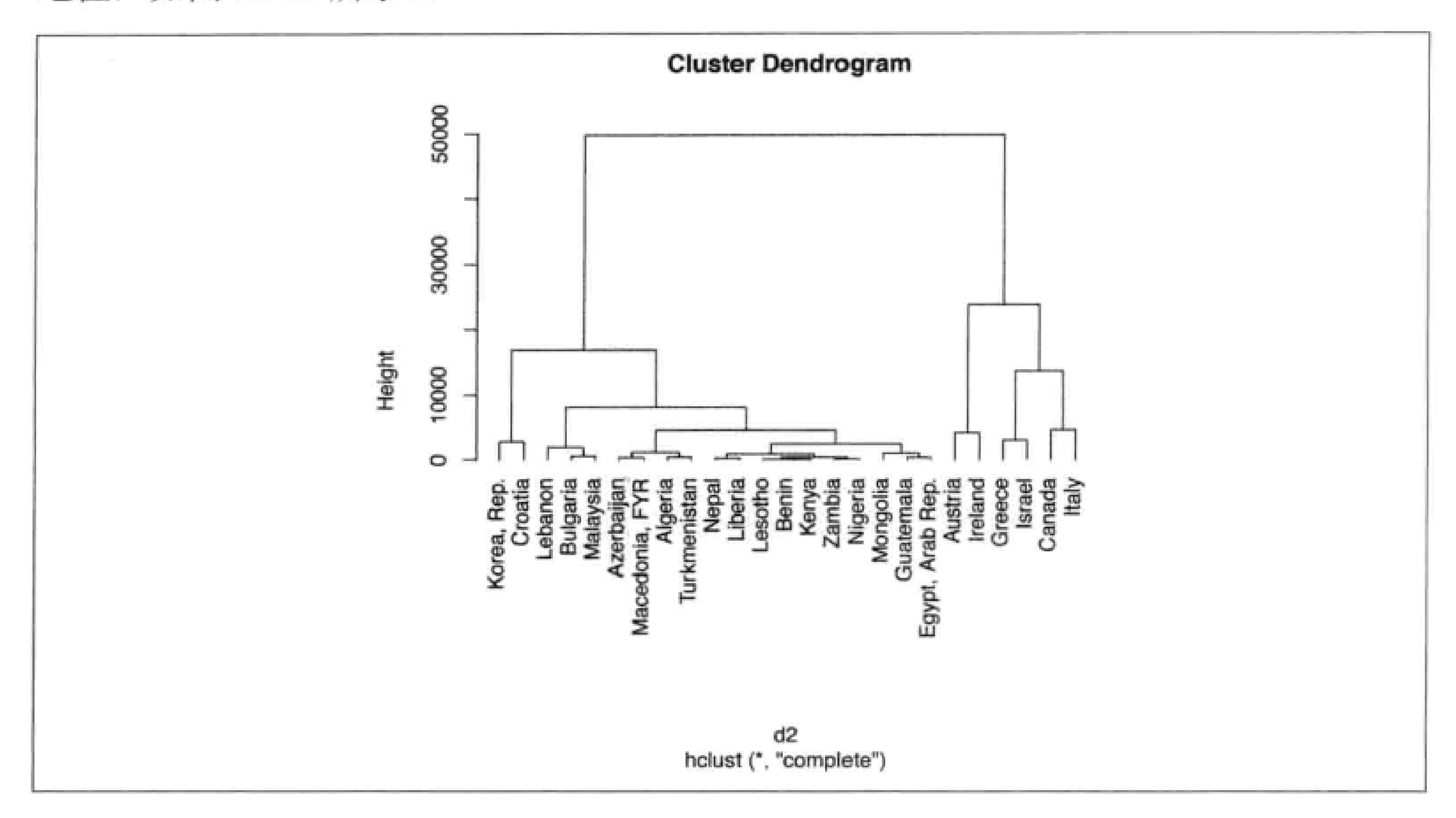


图 13-20　未标准化数据得到的谱系图（注意 `Height` 的值很大，这主要是没有标准化的 GDP 造成的）

对于距离的计算，我们使用的是默认的方法——`"euclidean"`，这个方法计算的是点与点之间的欧氏距离；还有其他的方法，比如 `"maximum"`，`"manhattan"`，`"canberra"`，`"binary"` 和 `"minkowski"`。

`hclust()` 函数提供了几种做聚类分析的方法，默认的方法是 `"complete"`；其他可用的方法包括 `"ward"`，`"single"`，`"average"`，`"mcquitty"`，`"median"` 和 `"centroid"`。

另见

输入 `?hclust` 来获取更多关于不同聚类方法的信息。

13.12　绘制向量场

问题

如何绘制一个向量场（vector field）？

方法

使用 `geom_segment()` 函数。在这个例子中，我们用的是 `isabel` 数据集。

```
library(gcookbook) # 数据集
isabel
```

	x	y	z	vx	vy	vz	t	speed

```
-83.00000 41.70000  0.035        NA        NA          NA        NA       NA
-83.00000 41.62786  0.035        NA        NA          NA        NA       NA
-83.00000 41.55571  0.035        NA        NA          NA        NA       NA
...
-62.04208 23.88036 18.035 -12.54371 -5.300128 -0.045253485 -66.96269 13.61749
-62.04208 23.80822 18.035 -12.56157 -5.254994 -0.020277001 -66.98840 13.61646
-62.04208 23.73607 18.035 -12.78071 -5.259613  0.005555035 -67.00575 13.82064
```

x 和 y 分别是经度和纬度，z 是高度，单位是千米。vx、vy、vz 是风速分量在各个方向的取值，单位是米每秒，speed 是风速。

高度（z）范围是 0.035 ～ 18.035 km。在本例中，只使用了最低高度的切片数据。

我们使用 geom_segment() 来绘制这些向量（见图 13-21）。每段都有一个起点和一个终点。我们用 x 和 y 值作为每段的起点，然后在此基础上分别加上 vx 和 vy 的一个比例，以此作为终点。如果不按比例缩小这些取值，线条就会变得很长：

```
islice <- subset(isabel, z == min(z))

ggplot(islice, aes(x=x, y=y)) +
       geom_segment(aes(xend = x + vx/50, yend = y + vy/50),
                    size = 0.25) # 线段 0.25 mm 粗
```

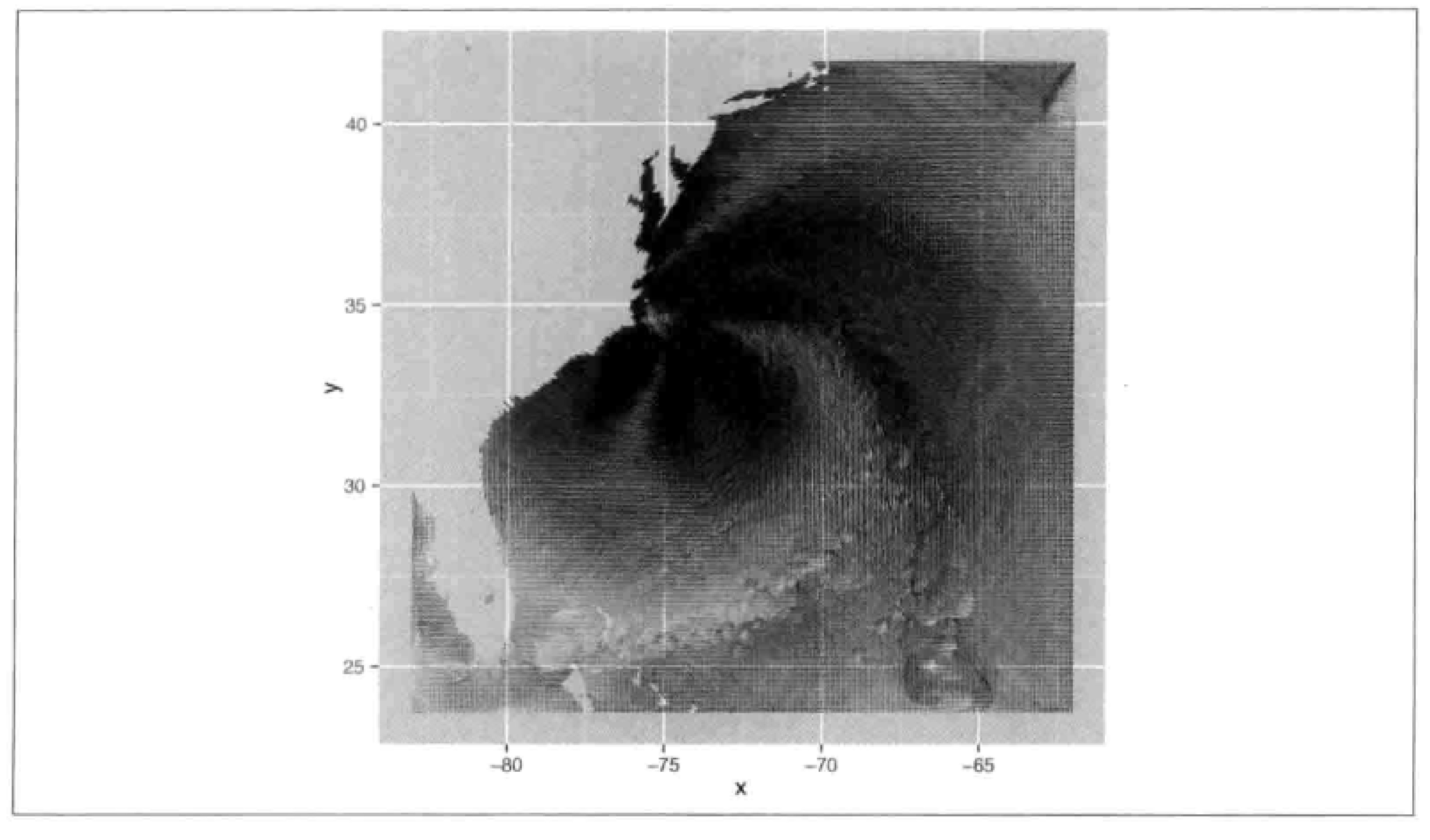


图 13-21　向量图的首次尝试——分辨率太高，但相对于较低的分辨率，它揭示出了一些有趣的模式

这个向量场有两个问题：数据分辨率太高不容易阅读，而且每段没有箭头表示方向。为了降低数据的分辨率，定义一个函数every_n()，在数据的每n个值中保留一个，其他的去掉。

```
# 选择 z 取值等于 z 的最小值的部分数据
islice <- subset(isabel, z == min(z))
```

```
# 向量 x 中每 'by' 个只里面保留一个
every_n <- function(x, by = 2) {
    x <- sort(x)
    x[seq(1, length(x), by = by)]
}

# x 和 y 每四个值保留一个
keepx <- every_n(unique(isabel$x), by=4)
keepy <- every_n(unique(isabel$y), by=4)

# 保留那些 x 值在 keepx 中并且 y 值在 keepy 中的数据
islicesub <- subset(islice, x %in% keepx & y %in% keepy)
```

现在我们已经得到了数据的一个子集，可以画出带箭头的图形，如图 13-22 所示：

```
# 使用 arrow()，需要加载 grid 包
library(grid)

# 用子集画图，箭头的长度为 0.1 cm
ggplot(islicesub, aes(x=x, y=y)) +
    geom_segment(aes(xend = x+vx/50, yend = y+vy/50),
                 arrow = arrow(length = unit(0.1, "cm")), size = 0.25)
```

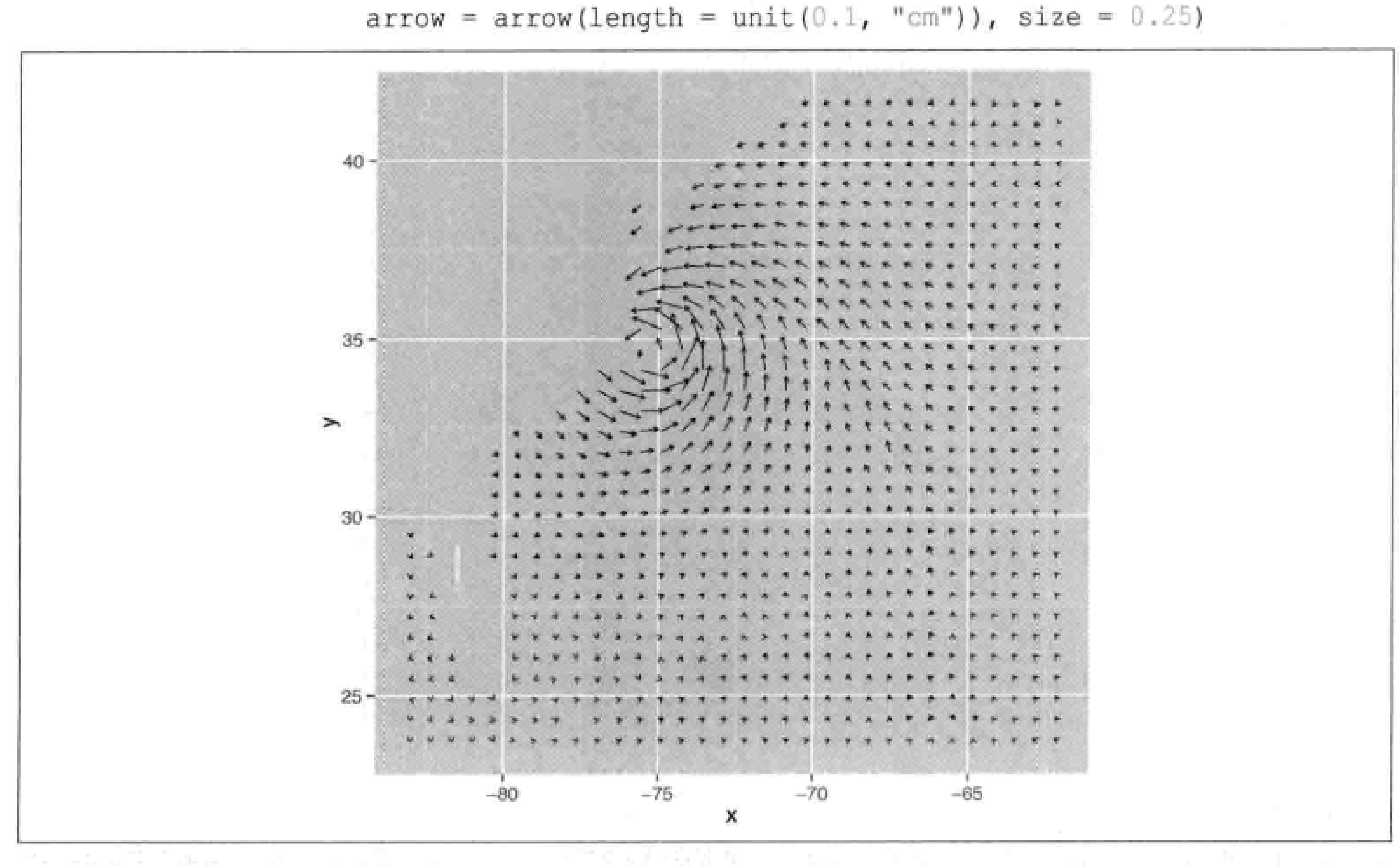


图 13-22　带箭头的向量图

讨论

箭头的一个影响是，短的向量会表现得比它实际长度的比例更大，这会导致曲解数据，为了减轻这种影响，把速度映射到其他属性上可能是有用的，如 size（线的粗细）、alpha

（透明度）或 colour（颜色）。这里，把速度映射到透明度 alpha 上（见图 13-23 左图）：

```
# 'speed' 列包含 z 的部分，计算水平速度
islicesub$speedxy <- sqrt(islicesub$vx^2 + islicesub$vy^2)

# 映射速度到透明度 alpha
ggplot(islicesub, aes(x=x, y=y)) +
    geom_segment(aes(xend = x+vx/50, yend = y+vy/50, alpha = speed),
                 arrow = arrow(length = unit(0.1,"cm")), size = 0.6)
```

下面，我们把速度映射到颜色 colour 上。我们还加入美国的部分地图并且使用 coord_cartesian()（没有这部分，会展示整个美国地图）放大感兴趣的区域，如图 13-23 右图所示：

```
# 得到美国地图数据
usa <- map_data("usa")

# 把数据映射到颜色上，颜色从 "grey80" 到 "darkred"
ggplot(islicesub, aes(x=x, y=y)) +
    geom_segment(aes(xend = x+vx/50, yend = y+vy/50, colour = speed),
                 arrow = arrow(length = unit(0.1,"cm")), size = 0.6) +
    scale_colour_continuous(low="grey80", high="darkred") +
    geom_path(aes(x=long, y=lat, group=group), data=usa) +
    coord_cartesian(xlim = range(islicesub$x), ylim = range(islicesub$y))
```

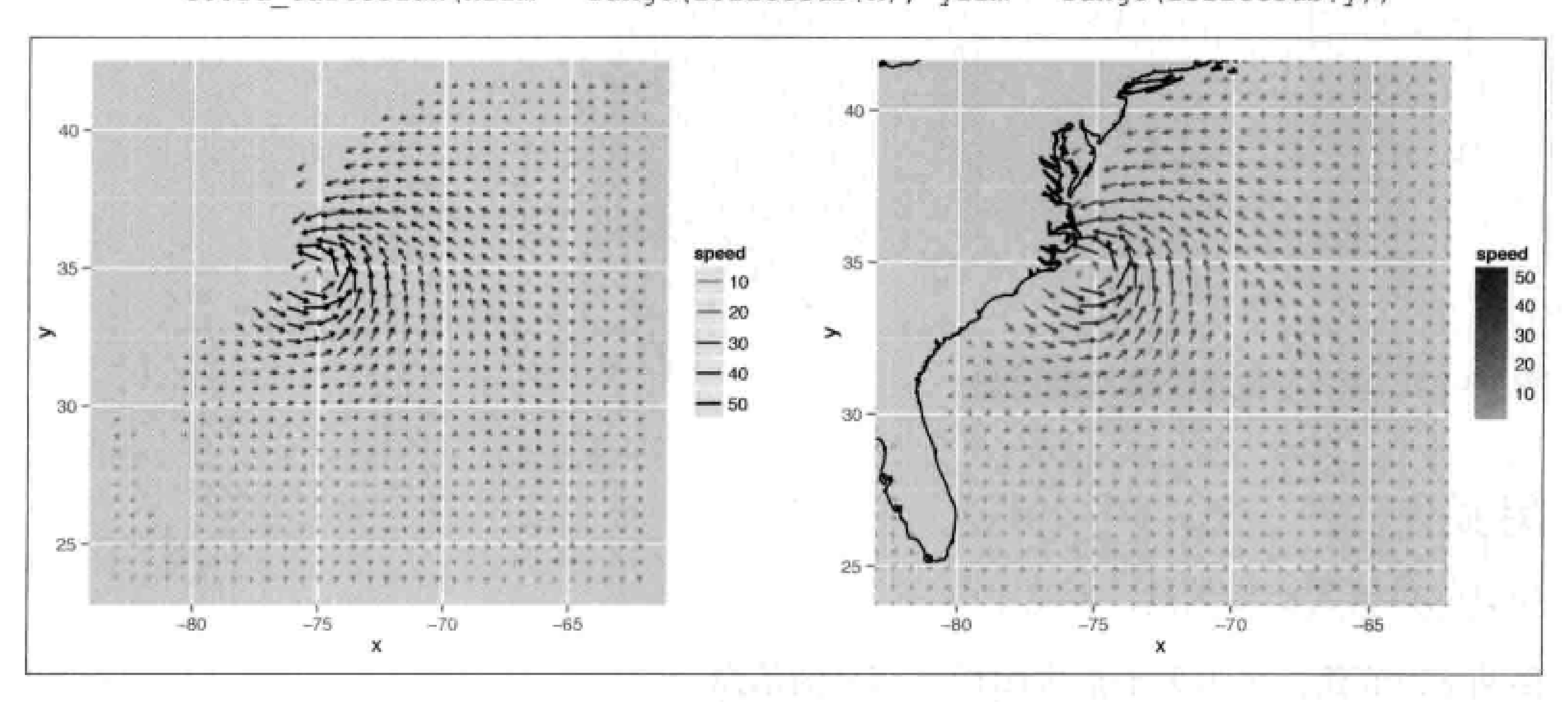


图 13-23　左图：速度映射到透明度上的向量场　右图：速度映射到颜色上的向量场

isabel 数据集是一个三维数据，所以我们可以画一个分面的图形，如图 13-24 所示。因为每个分面都很小，所以要用比之前更稀疏的子集。

```
# x 和 y 中每 5 个值保留 1 个，z 中每两个值保留一个
keepx <- every_n(unique(isabel$x), by=5)
keepy <- every_n(unique(isabel$y), by=5)
keepz <- every_n(unique(isabel$z), by=2)

isub <- subset(isabel, x %in% keepx & y %in% keepy & z %in% keepz)
```

```
ggplot(isub, aes(x=x, y=y)) +
    geom_segment(aes(xend = x+vx/50, yend = y+vy/50, colour = speed),
                 arrow = arrow(length = unit(0.1,"cm")), size = 0.5) +
    scale_colour_continuous(low="grey80", high="darkred") +
    facet_wrap( ~ z)
```

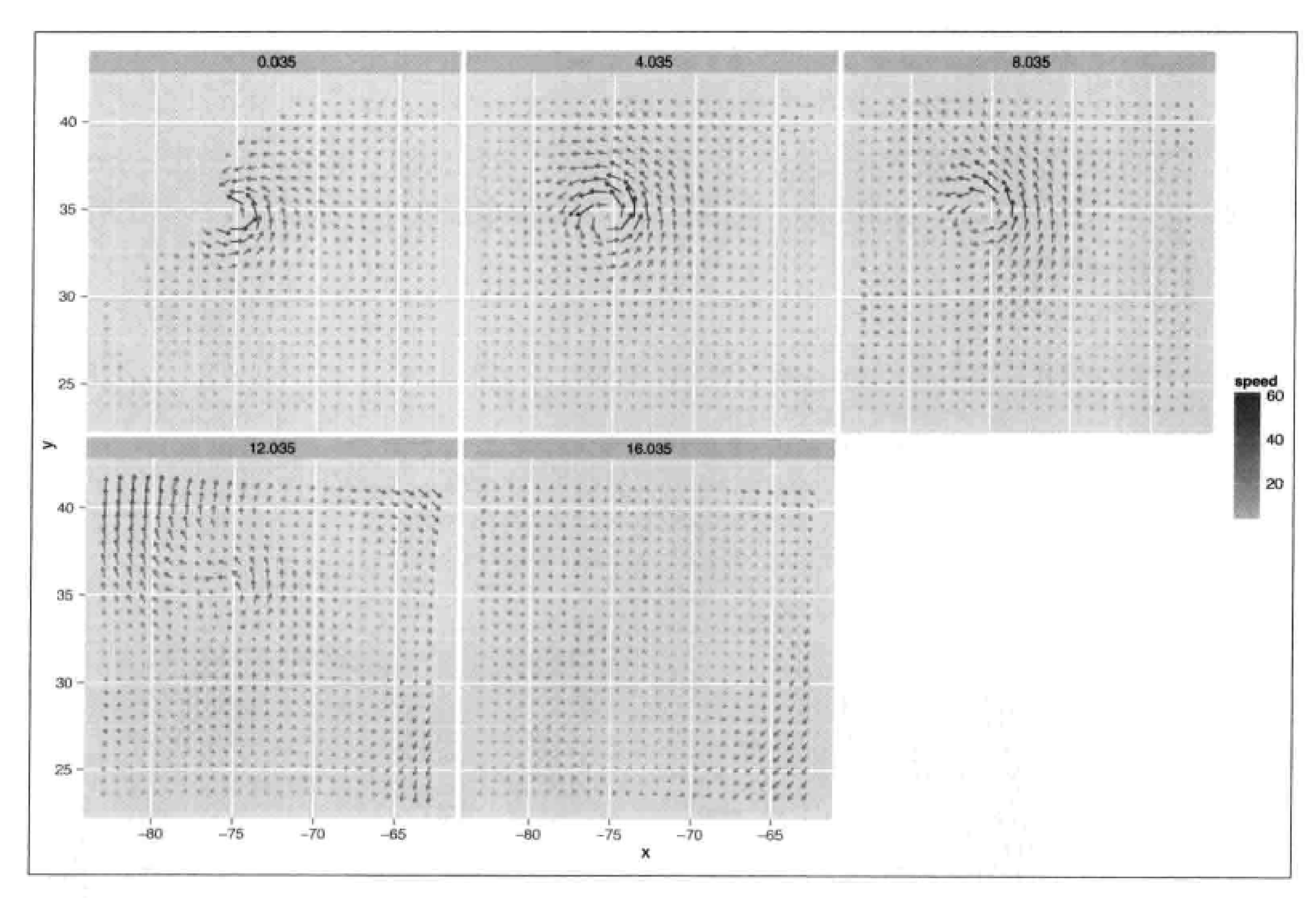


图 13-24　以 *z* 分面的风速向量场图

另见

如果你想用其他调色板，参见 12.6 节。

参阅 8.2 节获得更多关于扩大图形某部分的信息。

13.13　绘制 QQ 图

问题

如何绘制 QQ 图来比较经验分布和理论分布？

方法

使用 qqnorm() 和正态分布比较。给 qqnorm() 一个数值向量，在此基础上用 qqline()

加上理论分布（见图 13-25）：

```
library(gcookbook) # 为了使用数据集

# height 的 QQ 图
qqnorm(heightweight$heightIn)
qqline(heightweight$heightIn)

# age 的 QQ 图
qqnorm(heightweight$ageYear)
qqline(heightweight$ageYear)
```

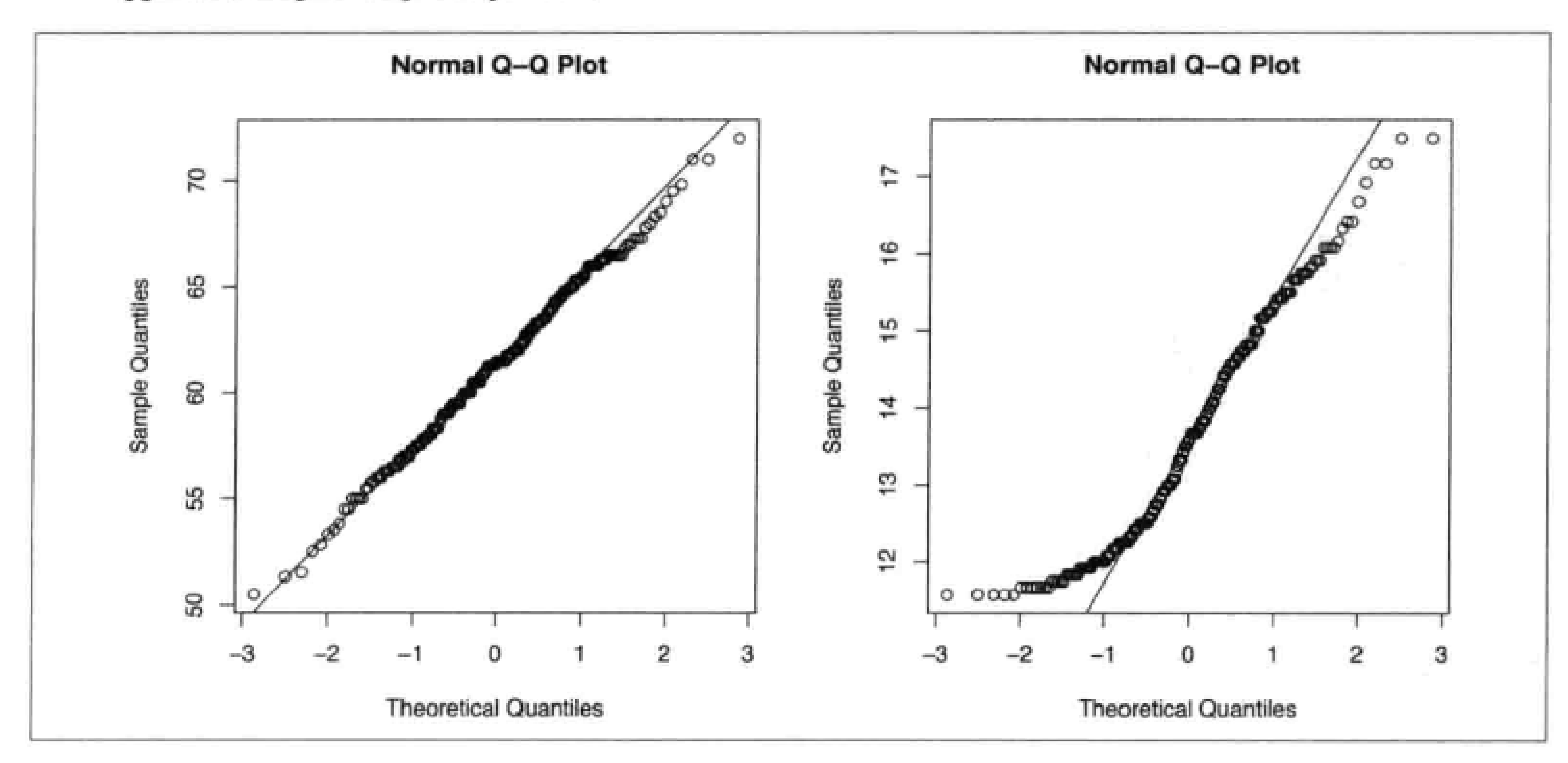


图 13-25　左图：**height** 的 QQ 图，和正态分布很接近　右图：**age** 的 QQ 图，不是正态分布

讨论

heightIn 的点很接近理论线，这意味着它的分布很接近正态分布。相反，ageYear 的点远离理论线，特别是在左面，这表明分布是有偏离的。此外，直方图在探索数据的分布上也是有帮助的。

另见

查看 ?qqplot 获得更多将数据和其他（非正态）理论分布进行比较的信息。

ggplot2 有一个 stat_qq() 函数，但是没有提供画 QQ 线的简单方法。

13.14　绘制经验累积分布函数图

问题

如何画一个数据集的经验累积分布函数图（ECDF）？

方法

使用 stat_ecdf()（见图 13-26）：

```
library(gcookbook) # 为了使用数据集

# heightIn 的 ecdf
ggplot(heightweight, aes(x=heightIn)) + stat_ecdf()

# ageYear 的 ecdf
ggplot(heightweight, aes(x=ageYear)) + stat_ecdf()
```

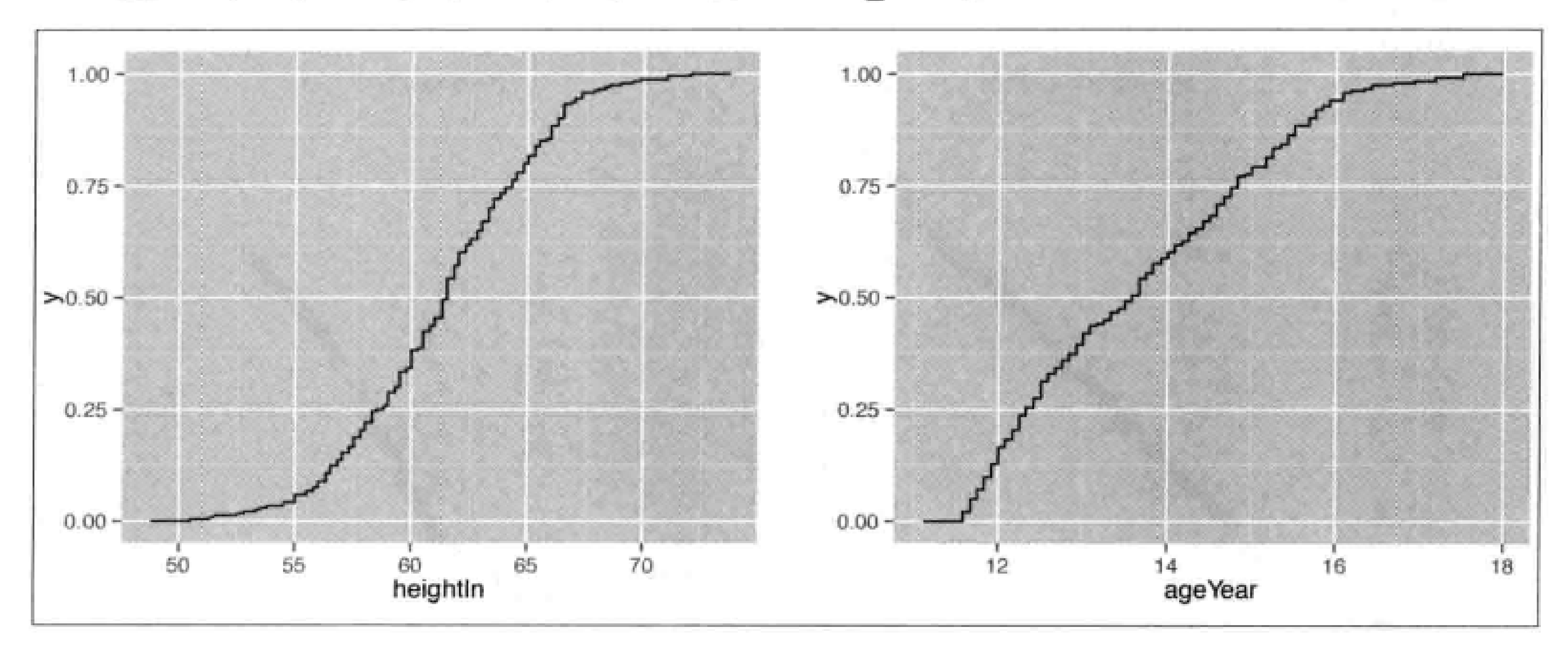


图 13-26　左图：**height** 变量的 ECDF　右图：**age** 变量的 ECDF

讨论

ECDF 表明了在观测数据中，小于或等于给定 *x* 值的观测所占的比例。因为是经验的，所以累积分布线在每个有一个或者更多观测值的 *x* 值处产生一个阶梯。

13.15　创建马赛克图

问题

如何绘制一个马赛克图来可视化一个列联表？

方法

使用 vcd 包里的 mosaic() 函数。在这个例子中，我们使用的是 UBAdmission 数据集，这个数据集是一个三维的列联表。首先用不同的方法观察一下这个数据集：

```
UCBAdmissions

, , Dept = A

            Gender
```

```
Admit       Male Female
  Admitted  512     89
  Rejected  313     19

 ... [four other Depts]

, , Dept = F

           Gender
Admit       Male Female
  Admitted   22     24
  Rejected  351    317

# 显示"平铺"后的列联表
ftable(UCBAdmissions)

                Dept   A   B   C   D   E   F
Admit    Gender
Admitted Male        512 353 120 138  53  22
         Female       89  17 202 131  94  24
Rejected Male        313 207 205 279 138 351
         Female       19   8 391 244 299 317

dimnames(UCBAdmissions)

$Admit
[1] "Admitted" "Rejected"

$Gender
[1] "Male"   "Female"

$Dept
[1] "A" "B" "C" "D" "E" "F"
```

三个维度分别是 Admit、Gender 和 Dept。为了可视化这些变量之间的关系（见图 13-27），使用 mosaic() 函数，输入的公式要包含在分割数据中使用的变量。

```
# 可能需要先安装包，install.packages("vcd")
library(vcd)

# 按照先 Admit 然后 Gender 再 Dept 的顺序分割数据
mosaic( ~ Admit + Gender + Dept, data=UCBAdmissions)
```

注意，mosaic() 是按照变量提供的顺序来分割数据的：首先是录取状态，然后是性别，最后是系。从图中可以看出，被拒绝的申请者明显比被录取的多。除此之外，在被录取的人中，男性比女性多，但在被拒绝的人中，男性和女性的比例差不多。但是很难比较每个系的情况。不同的分割数据的顺序可能会揭示不一样的有趣的信息。

另外一个观察数据的方式是按照系别、性别和录取状态的顺序分割数据，如图 13-28 所示。这时，录取状态是最后一个分离的变量，所以当按照系别和性别分离数据之后，每组中录取和拒绝的单元正好在彼此的旁边。

```
mosaic( ~ Dept + Gender + Admit, data=UCBAdmissions,
    highlighting="Admit", highlighting_fill=c("lightblue", "pink"),
    direction=c("v","h","v"))
```

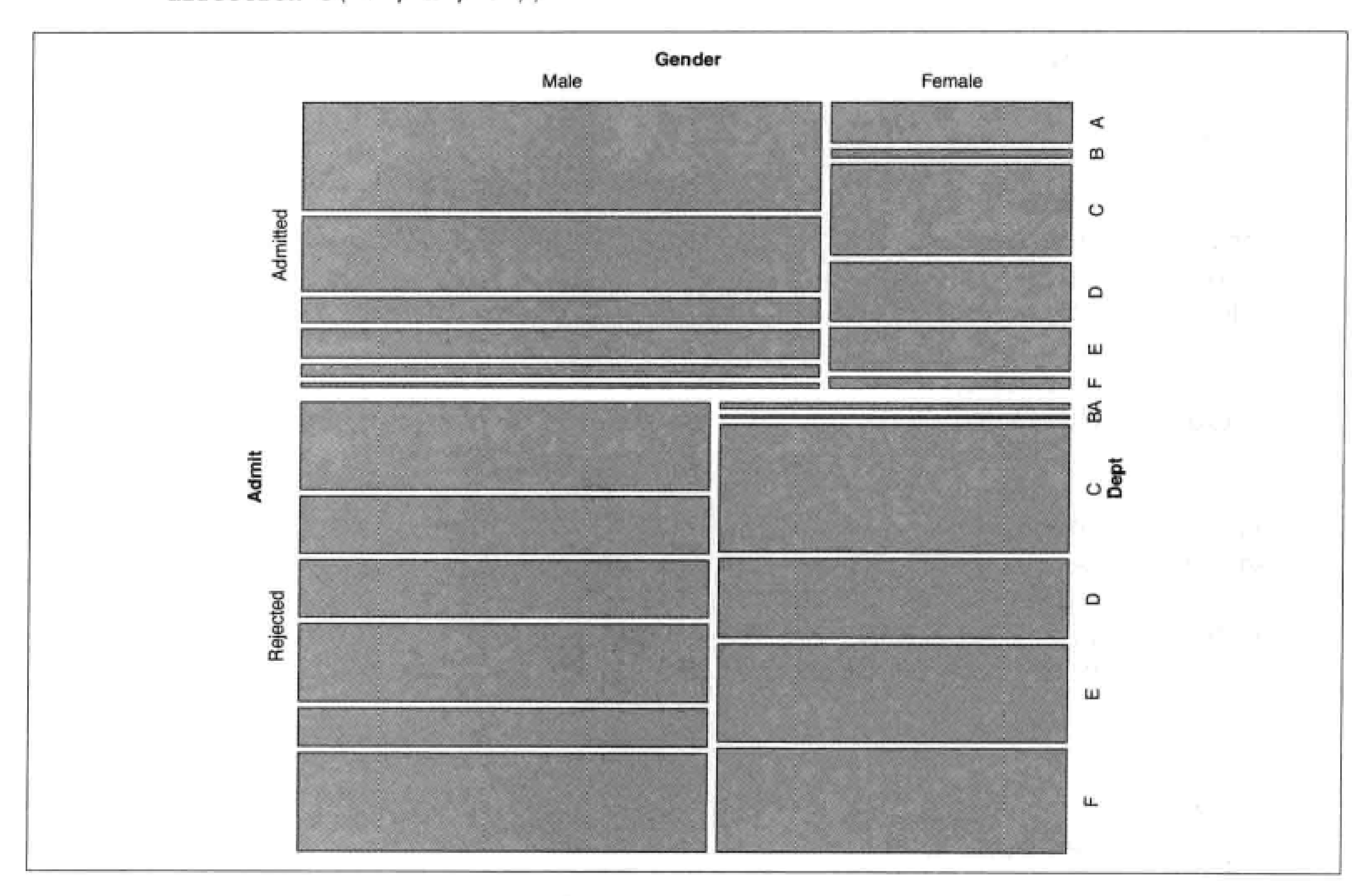


图 13-27　UC-Berkeley 招生数据的马赛克图——每个矩形的面积和对应的人数成正比

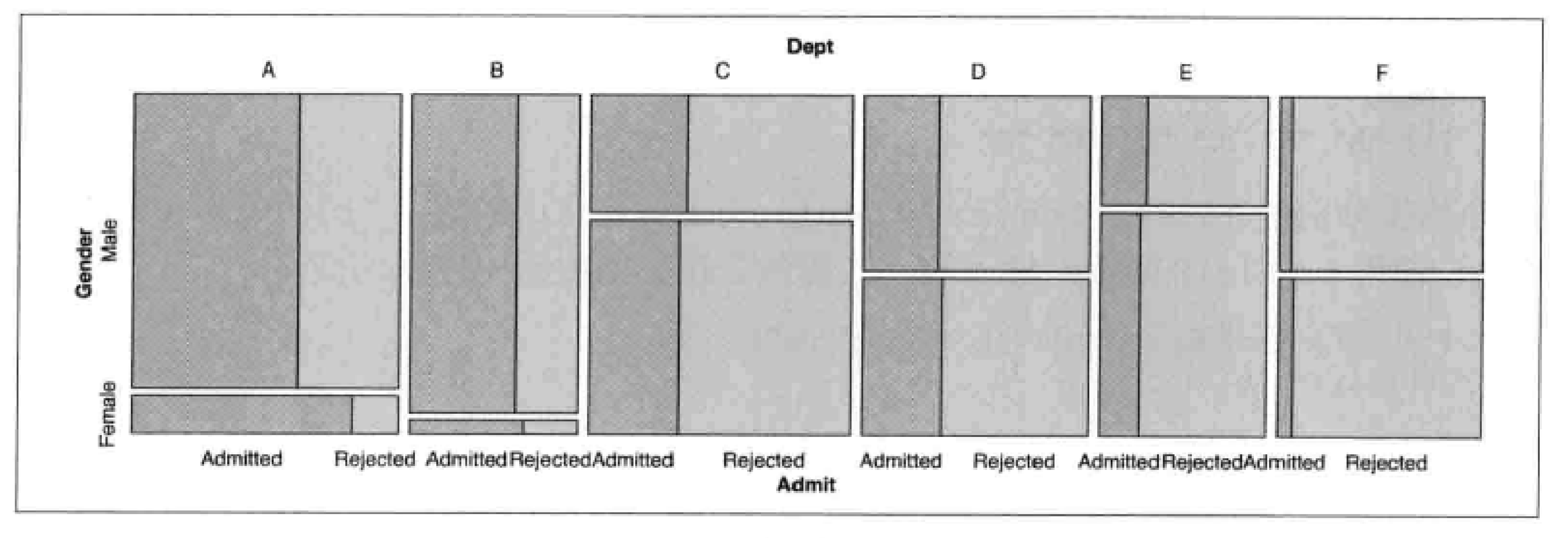


图 13-28　按另一种变量顺序分割数据的马赛克图：首先系别，然后性别，最后录取状态

我们还使用颜色指定了高亮的变量（Admit）。

讨论

在前面的例子中我们还为每个分割的变量指定了方向。其中，第一个变量 Dept 是垂直分割的，第二个变量 Gender 是水平分割的，第三个变量 Admit 是垂直分割的。我们选择这三个方向的原因是在这个例子中，这样做会比较容易比较每个系里面男女的差别。

我们还可以使用不同的分割方向，如图 13-29 和图 13-30 所示：

```
# 另一种可能的分割方向
mosaic( ~ Dept + Gender + Admit, data=UCBAdmissions,
    highlighting="Admit", highlighting_fill=c("lightblue", "pink"),
    direction=c("v", "v", "h"))

# 这个顺序比较男和女不是很容易
mosaic( ~ Dept + Gender + Admit, data=UCBAdmissions,
    highlighting="Admit", highlighting_fill=c("lightblue", "pink"),
    direction=c("v", "h", "h"))
```

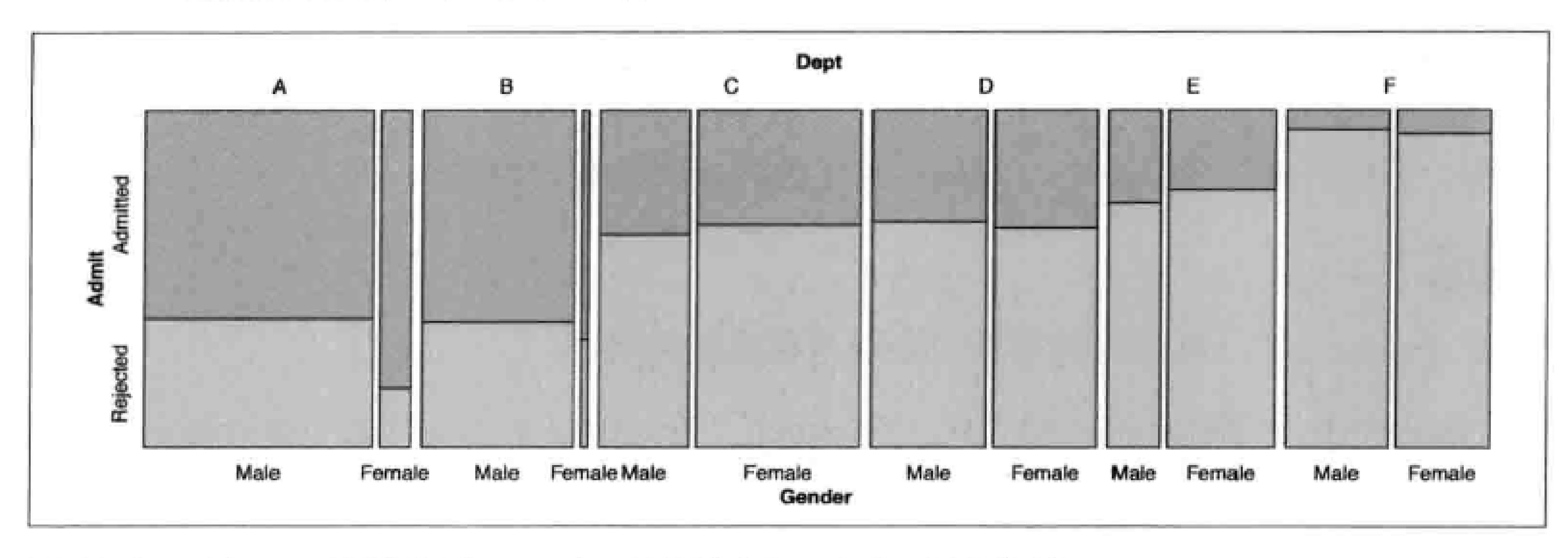


图 13-29　把 Dept 垂直分割，Gender 垂直分割，Admit 水平分割

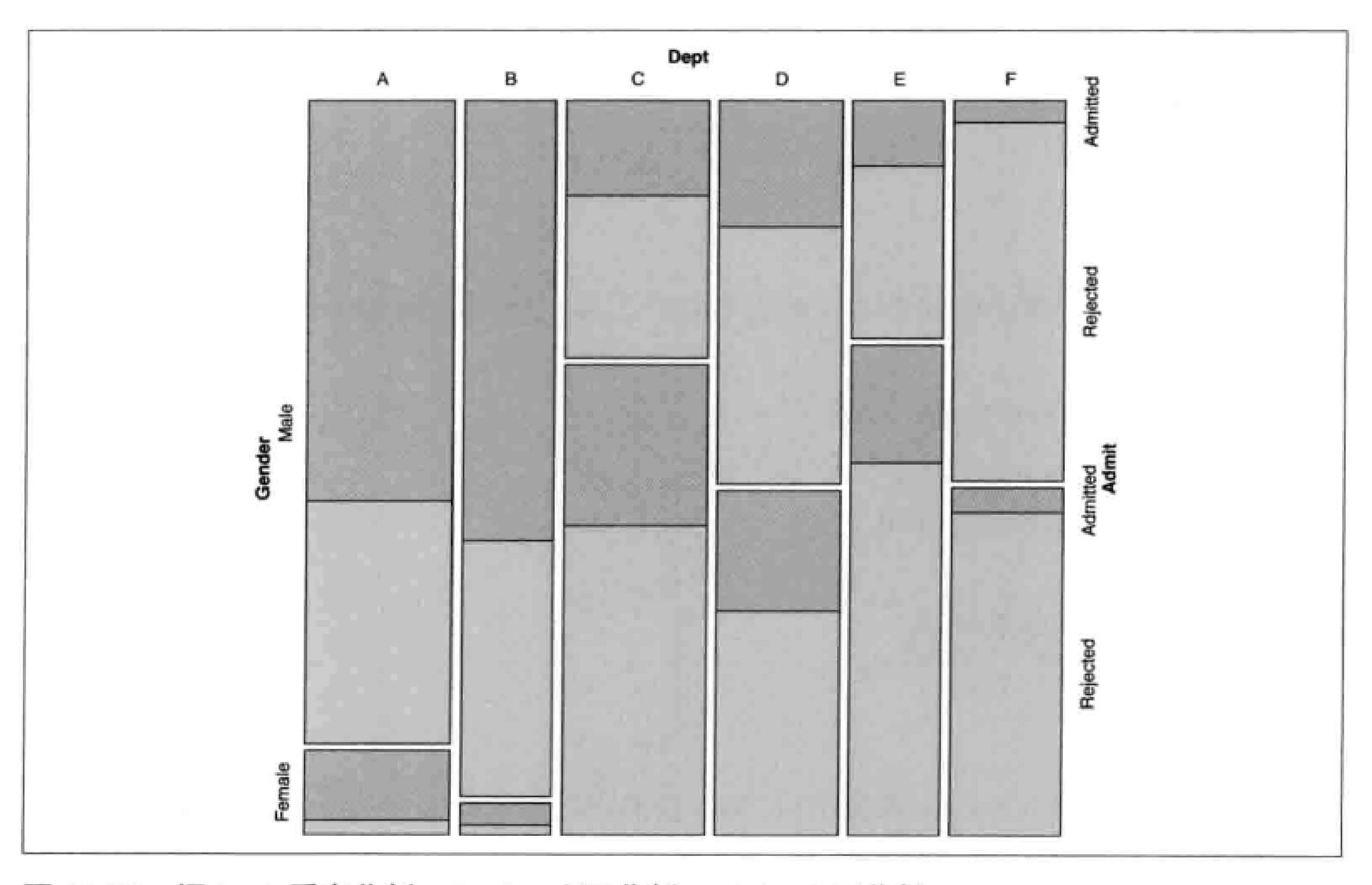


图 13-30　把 Dept 垂直分割，Gender 水平分割，Admit 水平分割

这里展示的例子是辛普森悖论中的一个案例，把组合并的时候，每个子组中的变量的关系会改变（或者相反）！UCBerkeley 表是加利福尼亚大学伯克利分校 1973 年的录取数据。总体来看，男性被录取的比率高于女性，由于这个原因，学校被起诉有性别偏见。但是当每个系分别检查的时候，发现男女的比率是差不多的。和总体录取比率不同的原因是女性更倾向于申请录取率低，竞争性强的系。

在图 13-28 和图 13-29 中，你可以发现在每个系中，男女的录取比率差不多。你也可以发现录取比率高的系（A 和 B）其申请者的性别比率非常不平衡：申请的男性比女性多得多。正如你所看到的，以不同的顺序和方向分割数据会得到数据不同层面的展示。在图 13-29 中，类似图 13-28，可以容易地比较每个系里和跨系之间男女的录取率。在图 13-30 中，把 Dept 垂直分割，Gender 水平分割，Admit 垂直分割，很难比较每个系里男女的录取比率，但是容易比较跨系的男女申请比率。

另见

查看 ?mosaicplot 可以得到另外一个画马赛克图的函数。

P.J. Bickel, E.A. Hammel, and J.W. O'Connell, "Sex Bias in Graduate Admissions: Data from Berkeley," Science 187 (1975): 398-404.

13.16 绘制饼图

问题

如何绘制一个饼图？

方法

使用 pie() 函数。在这个例子中（见图 13-31），我们使用来自 MASS 库里面的 survey 数据集。

```
library(MASS) # 为了使用数据集

# 得到 fold 变量每个水平的频数
fold <- table(survey$Fold)
fold

 L on R Neither  R on L
     99      18     120

# 画饼图
pie(fold)
```

我们在 pie() 中输入 table 类型的对象。我们也可以提供一个命名的向量，或者一个只有数值的向量和一个标签向量，如下所示：

```
pie(c(99, 18, 120), labels=c("L on R", "Neither", "R on L"))
```

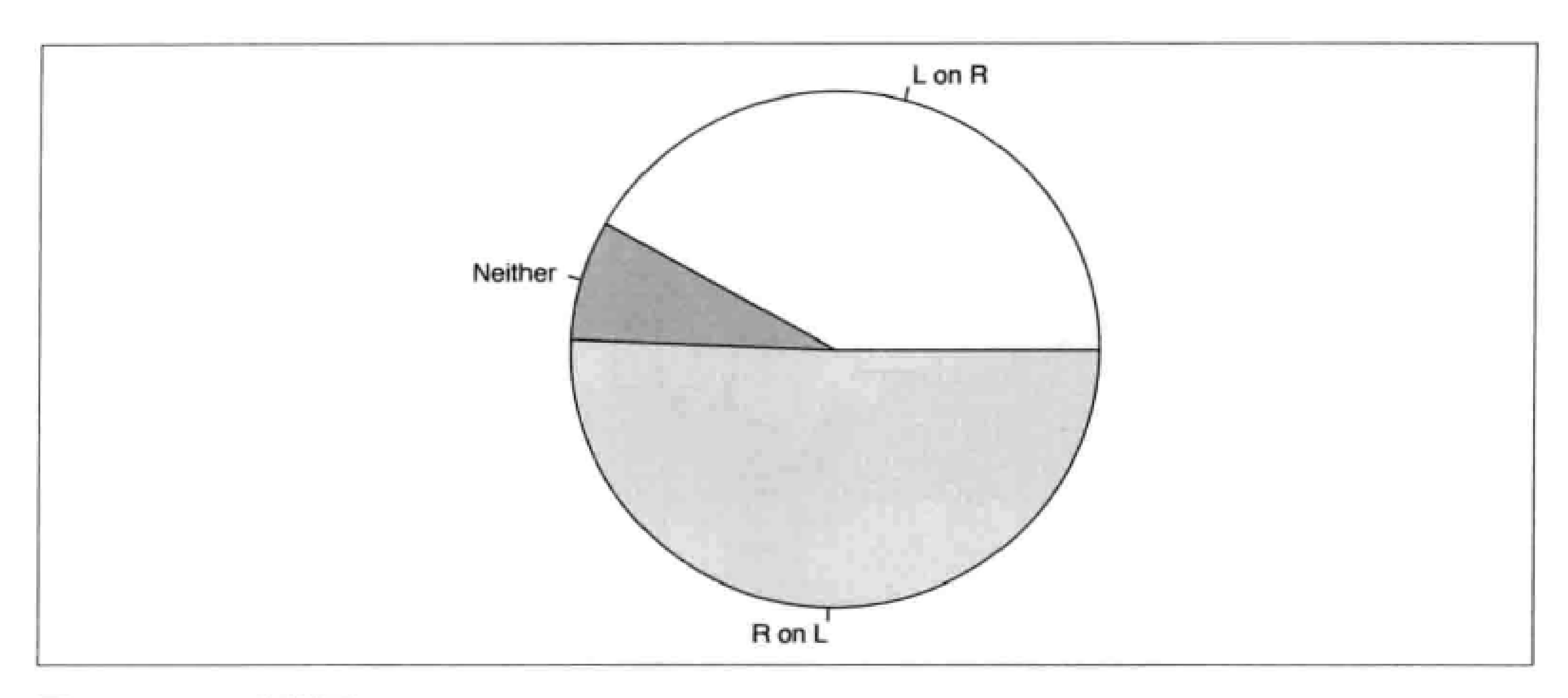


图 13-31　一个饼图

讨论

饼图的滥用是可视化专家经常指责的一个话题。如果你考虑使用饼图，不妨考虑使用条形图（或者堆积条形图），它们也许可以更加有效地传达信息。尽管有缺点，但是饼图有一个非常重要的优点，就是每个人都能够读懂它。

13.17　创建地图

问题

如何绘制一个地图？

方法

从 maps 包里面获取地图数据，用 geom_polygon()（可以用颜色填充）或者 geom_path()（不能填充）绘制。经度和纬度默认是画在直角坐标系中的，但是你可以用 coord_map() 指定一个投影。默认的投影是 "mercator"（墨卡托投影），和直角坐标系不一样，墨卡托投影中纬度线之间的距离会逐渐发生变化（见图 13-32）。

```
library(maps) # 为了使用地图数据
# 美国地图数据
states_map <- map_data("state") # 必须载入 ggplot2 以使用 map_data()

ggplot(states_map, aes(x=long, y=lat, group=group)) +
    geom_polygon(fill="white", colour="black")

# geom_path (没有填充) 和墨卡托投影①
ggplot(states_map, aes(x=long, y=lat, group=group)) +
    geom_path() + coord_map("mercator")
```

① 需要 mapproj 包。——译者注

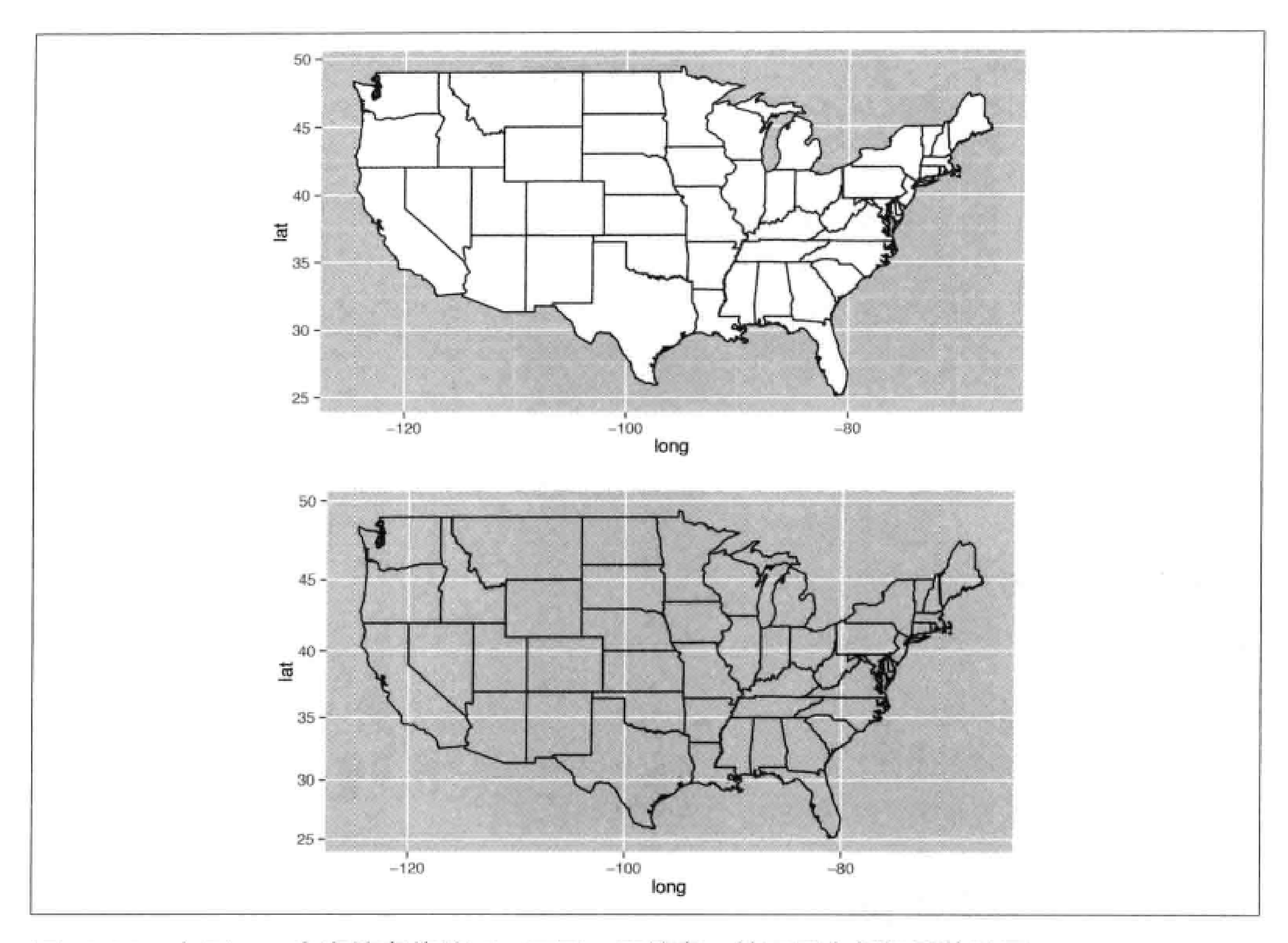


图 13-32　上图：一个有填充的地图　下图：无填充，使用墨卡托投影的地图

讨论

map_data() 函数返回的是一个含有如下几列的数据框。

- long：纬度。
- lat：经度。
- group：这是每个多边形的分组变量。一个区域或子区域可能有多个多边形，例如：如果它含有岛屿。
- order：在一组里面每个点的连接顺序。
- region：基本是国家和地区的名字，也有其他的对象（例如一些湖）。
- subregion：一个区域中子区域的名字，可能包含多个组别。例如 Alaska 子区域包含很多岛屿，每个都是一组。

还有很多不同的地图，包括 world、nz、france、italy、usa（美国的轮廓）、state（美国的每个州）和 county（美国的每个郡）。例如，世界地图的数据如下所示：

```
# 世界地图数据
world_map <- map_data("world")
world_map
```

```
        long       lat group order       region subregion
```

```
-133.3664 58.42416      1     1      Canada      <NA>
-132.2681 57.16308      1     2      Canada      <NA>
-132.0498 56.98610      1     3      Canada      <NA>
...
 124.7772 11.35419  2284 27634 Philippines     Leyte
 124.9697 11.30280  2284 27635 Philippines     Leyte
 125.0155 11.13887  2284 27636 Philippines     Leyte
```

如果你想画世界地图中某个没有单独地图数据的区域，可以首先查找区域的名字，如下所示：

```
sort(unique(world_map$region))

"Afghanistan"           "Albania"             "Algeria"
"American Samoa"        "Andaman Islands"     "Andorra"
"Angola"                "Anguilla"            "Antarctica"
 ...
"USA"                   "USSR"                "Vanuatu"
"Venezuela"             "Vietnam"             "Virgin Islands"
"Vislinskiy Zaliv"      "Wales"               "West Bank"
"Western Sahara"        "Yemen"               "Yugoslavia"
"Zaire"                 "Zambia"              "Zimbabwe"

# 你可能发现地图有些过时了！
```

可以从世界地图中得到指定区域的地图数据（见图 13-33）。

```
euro <- map_data("world", region=c("UK", "France", "Netherlands","Belgium"))

# Map region to fill color
ggplot(euro, aes(x=long, y=lat, group=group, fill=region)) +
geom_polygon(colour="black") +
scale_fill_brewer(palette="Set2") +
scale_y_continuous(limits=c(40, 60))+
scale_x_continuous(limits=c(-25, 25))
```

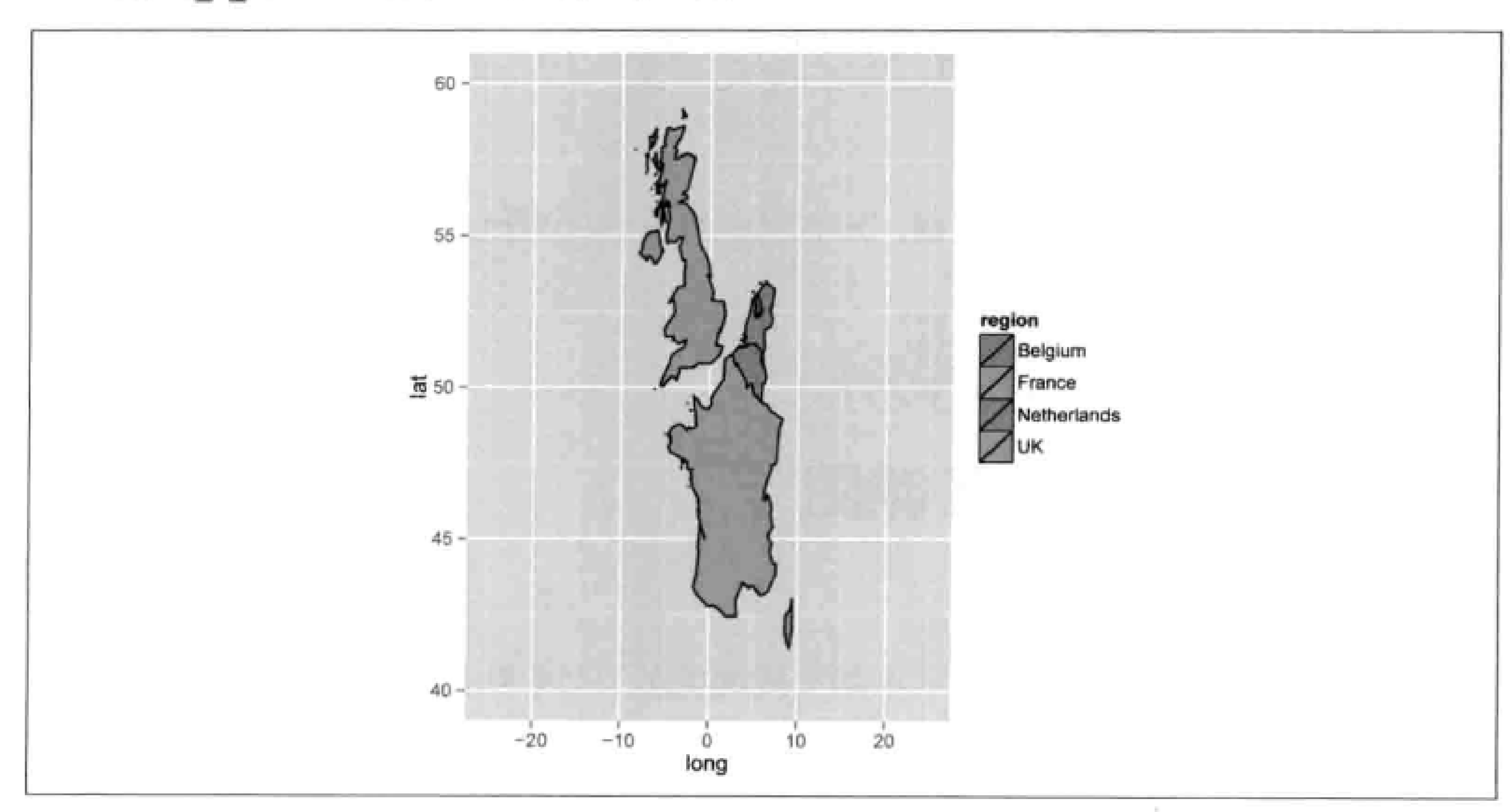


图 13-33　在世界地图中绘制指定区域的地图

如果一个区域有单独的地图，如 nz（新西兰），那么地图数据就会比从世界地图中提取出的数据的分辨率高，如图 13-34 所示：

```
# 从世界地图中得到新西兰地图数据
nz1 <- map_data("world", region="New Zealand")
nz1 <- subset(nz1, long > 0 & lat > -48) # 剔除岛屿
ggplot(nz1, aes(x=long, y=lat, group=group)) + geom_path()

# 从新西兰（nz）地图中得到新西兰地图数据
nz2 <- map_data("nz")
ggplot(nz2, aes(x=long, y=lat, group=group)) + geom_path()
```

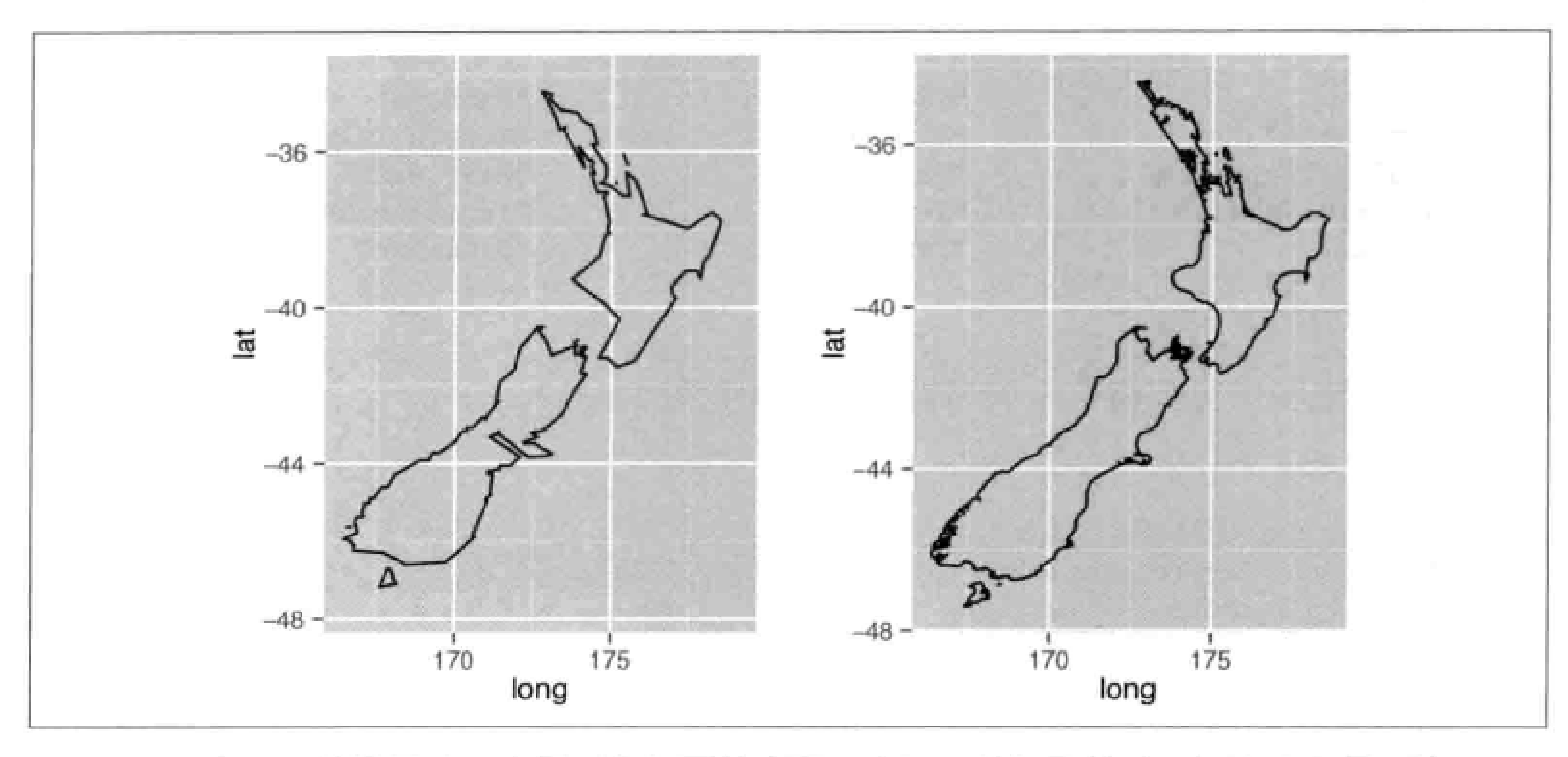


图 13-34　左图：从世界地图中得到的新西兰地图　右图：从新西兰（nz）地图中得到的

另见

查看 mapdata 包获得更多地图数据，还有高分辨率的世界地图 worldHires。

查看 map() 函数，快速产生地图。

查看 ?mapproject 获得地图的投影方法。

13.18　绘制等值区域图

问题

如何创建一个地图，使得不同区域根据变量值填充不同的颜色？

方法

把变量值和地图数据合并在一起，然后把一个变量映射到 fill 上。

```
# 把 USArrests 数据集转换成正确的格式
crimes <- data.frame(state = tolower(rownames(USArrests)), USArrests)
crimes

                        state Murder Assault UrbanPop Rape
Alabama               alabama   13.2     236       58 21.2
Alaska                 alaska   10.0     263       48 44.5
Arizona               arizona    8.1     294       80 31.0
   ...
West Virginia   west virginia    5.7      81       39  9.3
Wisconsin           wisconsin    2.6      53       66 10.8
Wyoming               wyoming    6.8     161       60 15.6

library(maps) # 为了使用地图数据
states_map <- map_data("state")

# 合并数据集
crime_map <- merge(states_map, crimes, by.x="region", by.y="state")

# 合并之后，顺序发生了变化，可能会导致多边形位置不对，所以要对数据排序
head(crime_map)

  region      long      lat group order subregion Murder Assault UrbanPop Rape
 alabama -87.46201 30.38968     1     1      <NA>   13.2     236       58 21.2
 alabama -87.48493 30.37249     1     2      <NA>   13.2     236       58 21.2
 alabama -87.95475 30.24644     1    13      <NA>   13.2     236       58 21.2
 alabama -88.00632 30.24071     1    14      <NA>   13.2     236       58 21.2
 alabama -88.01778 30.25217     1    15      <NA>   13.2     236       58 21.2
 alabama -87.52503 30.37249     1     3      <NA>   13.2     236       58 21.2

library(plyr) # 为了使用 arrange() 函数
# 按照 group, order 排序
crime_map <- arrange(crime_map, group, order)
head(crime_map)

  region      long      lat group order subregion Murder Assault UrbanPop Rape
 alabama -87.46201 30.38968     1     1      <NA>   13.2     236       58 21.2
 alabama -87.48493 30.37249     1     2      <NA>   13.2     236       58 21.2
 alabama -87.52503 30.37249     1     3      <NA>   13.2     236       58 21.2
 alabama -87.53076 30.33239     1     4      <NA>   13.2     236       58 21.2
 alabama -87.57087 30.32665     1     5      <NA>   13.2     236       58 21.2
 alabama -87.58806 30.32665     1     6      <NA>   13.2     236       58 21.2
```

当数据的格式正确时，就可以画出图形（见图 13-35），把其中一列数值映射到 fill 上：

```
ggplot(crime_map, aes(x=long, y=lat, group=group, fill=Assault)) +
    geom_polygon(colour="black") +
    coord_map("polyconic")
```

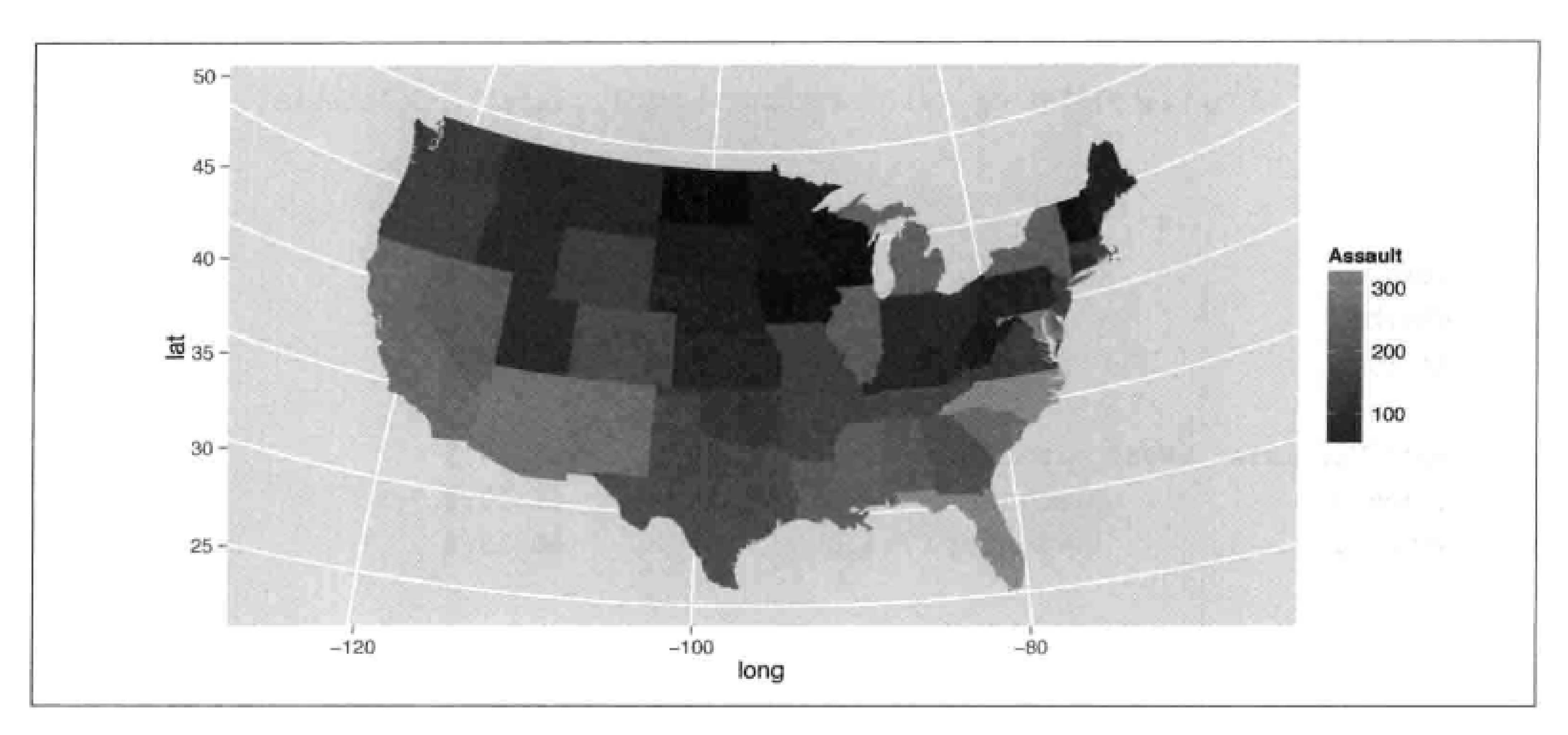


图 13-35　将一个变量映射到 fill 上的地图

讨论

前面的例子用的是默认的颜色标度，从深蓝渐变到浅蓝。如果你想展示变量值是如何从某个中点值向外发散的，可以用 scale_fill_gradient2()，如图 13-36 所示：

```
ggplot(crimes, aes(map_id = state, fill=Assault)) +
    geom_map(map = states_map, colour="black") +
    scale_fill_gradient2(low="#559999", mid="grey90", high="#BB650B",
                         midpoint=median(crimes$Assault)) +
    expand_limits(x = states_map$long, y = states_map$lat) +
    coord_map("polyconic")
```

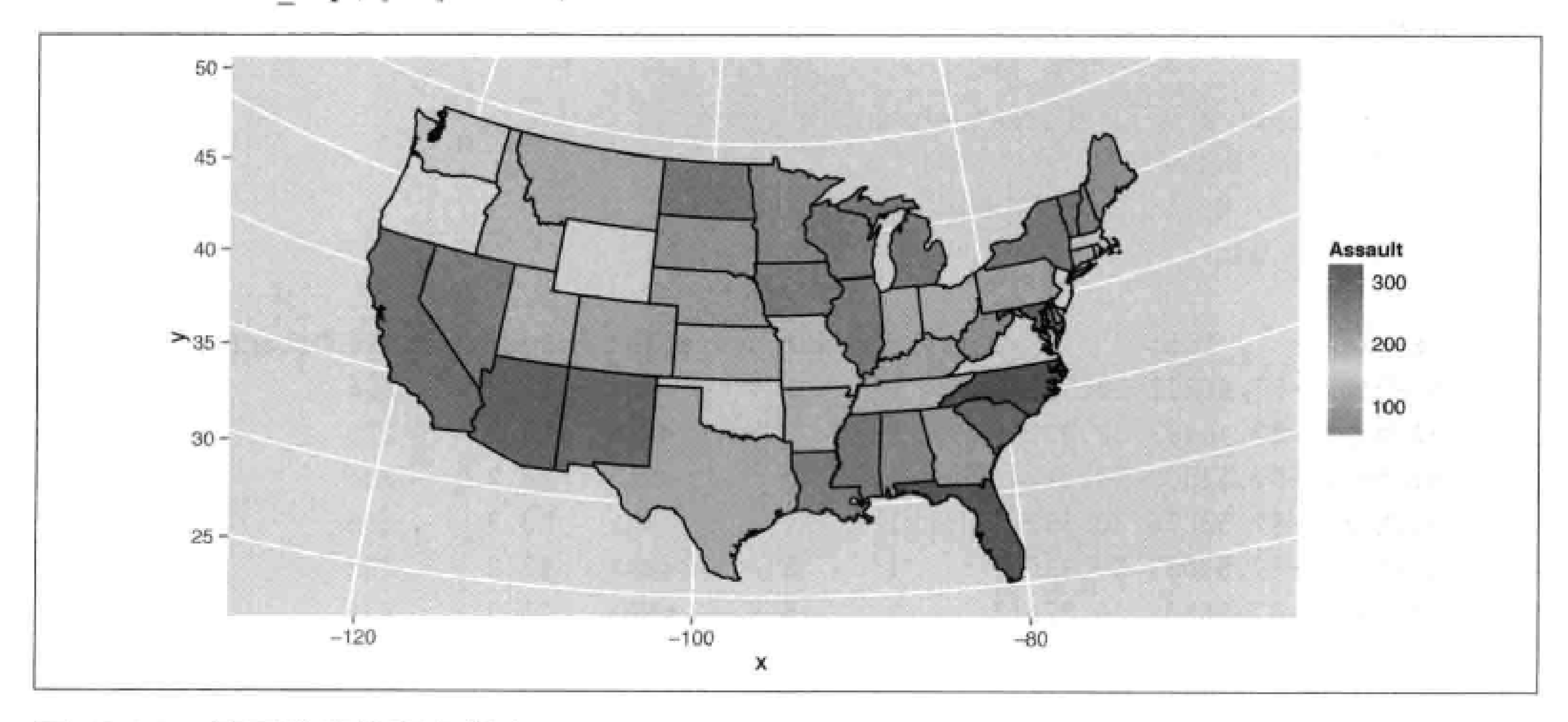


图 13-36　使用发散的颜色标度

前面的例子把连续取值映射到 fill 上，但是我们也可以用离散取值。有些时候离散的取值更容易解释。例如，我们可以把取值按照分位数进行分类。展示分位数，如图 13-37 所示：

```
# 找到分位数的边界
qa <- quantile(crimes$Assault, c(0, 0.2, 0.4, 0.6, 0.8, 1.0))
qa

   0%   20%   40%   60%   80%  100%
 45.0  98.8 135.0 188.8 254.2 337.0

# 加入一个分位数类别的列
crimes$Assault_q <- cut(crimes$Assault, qa,
                        labels=c("0-20%", "20-40%", "40-60%", "60-80%", "80-100%"),
                        include.lowest=TRUE)
crimes

                    state Murder Assault UrbanPop Rape Assault_q
Alabama           alabama   13.2     236       58 21.2    60-80%
Alaska             alaska   10.0     263       48 44.5   80-100%
 ...
Wisconsin       wisconsin    2.6      53       66 10.8     0-20%
Wyoming           wyoming    6.8     161       60 15.6    40-60%

# 产生一个有 5 个离散取值的调色板
pal <- colorRampPalette(c("#559999", "grey80", "#BB650B"))(5)
pal

 "#559999" "#90B2B2" "#CCCCCC" "#C3986B" "#BB650B"

ggplot(crimes, aes(map_id = state, fill=Assault_q)) +
    geom_map(map = states_map, colour="black") +
    scale_fill_manual(values=pal) +
    expand_limits(x = states_map$long, y = states_map$lat) +
    coord_map("polyconic") +
    labs(fill="Assault Rate\nPercentile")
```

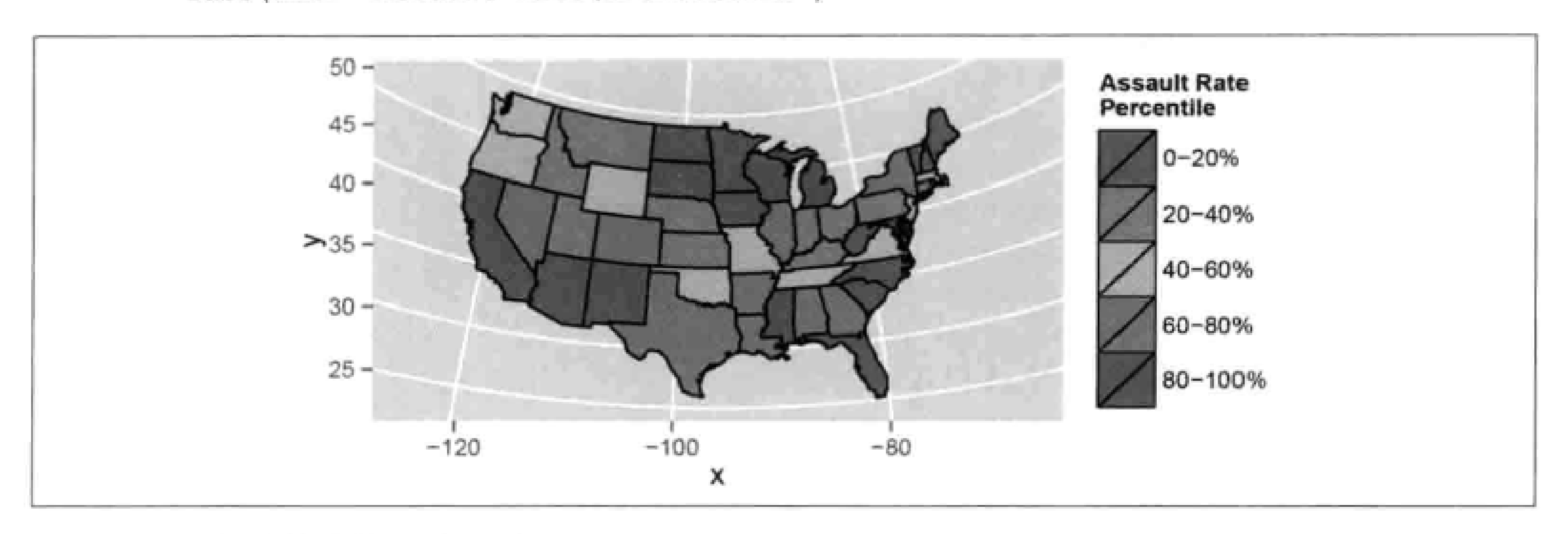


图 13-37　离散数值的等值区域图

另外一个画等值区域图（Choropleth Map）的方法是使用 geom_map()，它不需要把变量取值和地图数据合并起来。因此，这个方法可以比之前描述的方法更快地生成地图。

在这个方法中，地图数据框中必须有 lat（经度）、long（纬度）和 region（区域）列。

在数值数据框中，必须有一列能够和地图数据框的 region（区域）列匹配上，并且这一列要映射到 map_id 图形属性上。例如，下面的代码和前面的例子效果相同：

```
# crimes 中 'state' 列要和 states_map 中的 'region' 列匹配
ggplot(crimes, aes(map_id = state, fill=Assault)) +
    geom_map(map = states_map) +
    expand_limits(x = states_map$long, y = states_map$lat) +
    coord_map("polyconic")
```

注意，我们还需要用 expand_limits()，这是因为和大多数几何对象不一样，geom_map() 不能自动设定 x 和 y 的界限，使用 expand_limits() 可以使得经纬度包含 x 和 y 值（另外一种能够得到相同结果的方法是用 ylim() 和 xlim()）。

另见

将数据叠加在一个地图上的方法，参见 13.12 节。

更多使用连续颜色的方法，参见 12.6 节。

13.19 创建空白背景的地图

问题

何如把地图中的背景元素去掉？

方法

首先，保存下面的主题：

```
# 创建一个去掉了很多背景元素的主题
theme_clean <- function(base_size = 12) {
require(grid) # unit() 函数需要
  theme_grey(base_size) %+replace%
  theme(
    axis.title        = element_blank(),
    axis.text         = element_blank(),
    panel.background = element_blank(),
    panel.grid        = element_blank(),
    axis.ticks.length = unit(0, "cm"),
    axis.ticks.margin = unit(0, "cm"),
    panel.margin      = unit(0, "lines"),
    plot.margin       = unit(c(0, 0, 0, 0), "lines"),
    complete = TRUE
  )
}
```

然后把它加在地图上（见图 13-38）。在这个例子中，我们把它加在了 13.18 节中创建的一个等值区域图上。

```
ggplot(crimes, aes(map_id = state, fill=Assault_q)) +
    geom_map(map = states_map, colour="black") +
    scale_fill_manual(values=pal) +
    expand_limits(x = states_map$long, y = states_map$lat) +
    coord_map("polyconic") +
    labs(fill="Assault Rate\nPercentile") +
    theme_clean()
```

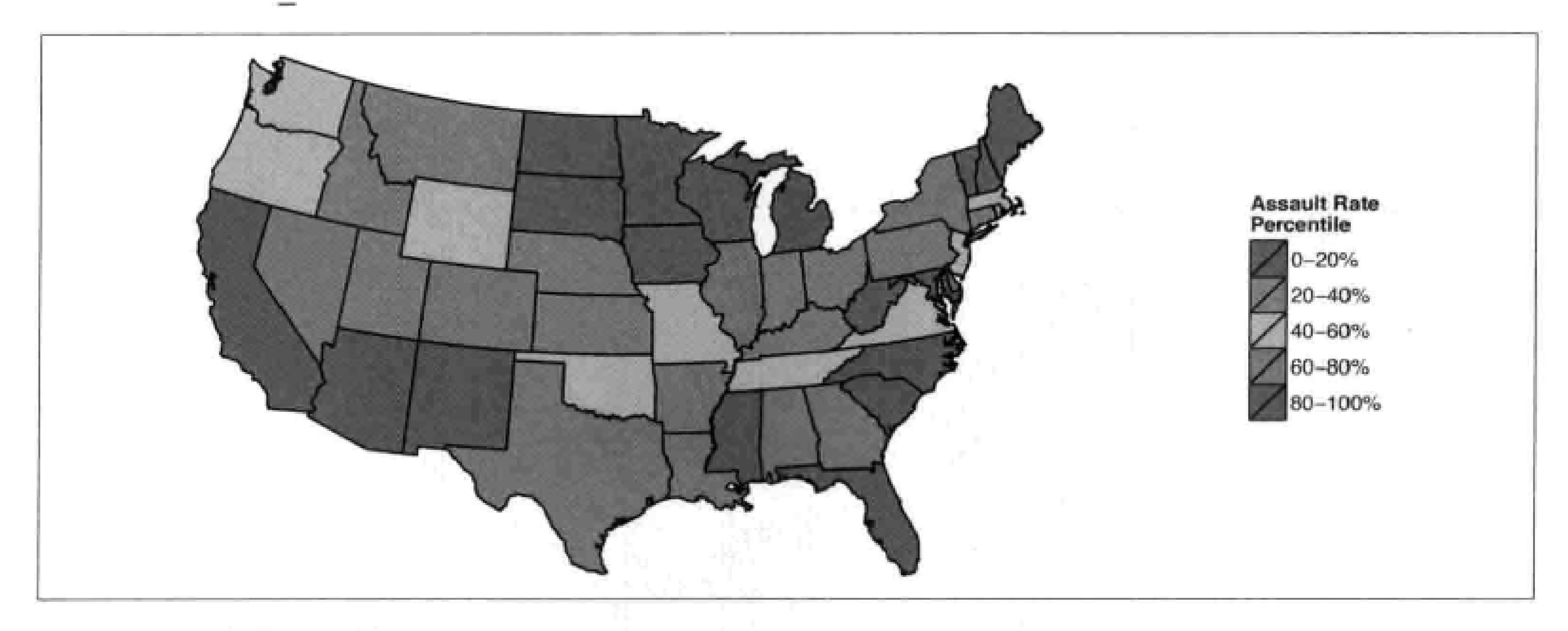
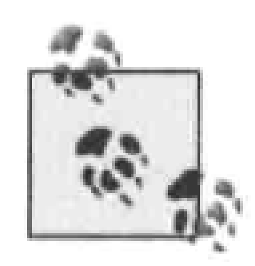


图 13-38　空白背景的地图



在 R 2.15.2 以及更早的版本中存在一个缺陷，可能会出现下面这样的错误：

```
Error in grid.Call.graphics(L_setviewport, pvp, TRUE) :
  Non-finite location and/or size for viewport
```

出现这样错误的原因是一些维度的和的长度是 0，grid 图形引擎不能很好地处理这个问题。这个缺陷应该在 R 3.0 中被修复了。如果你使用的 R 出现了这个问题，在改变主题的时候请使用 `axis.ticks.margin = unit(0.01, "cm")`，而不是 `axis.ticks.margin = unit(0,"cm")`。

讨论

在有些地图中，包含前后相关的信息是很重要的，例如经度和纬度。在另外一些地图中，这些信息却不再重要了——它们反而会分散所传达的信息。在图 13-38 中，观察者可能不关心各个州的经度和纬度。他们可以通过形状和相对位置判断这些州，并且即使他们无法判断，经度和纬度也没有什么帮助。

13.20　基于空间数据格式（shapefile）创建地图

问题

如何基于 Esri shapefile 创建地图？

方法

用 maptools 包中的 `readShapePoly()` 载入空间数据文件，用 `fortify()` 把数据转化成数据框的格式，然后画图（见图 13-39）：

```
library(maptools)

#  载入空间数据并转化成数据框
uk_shp <- readShapePoly("GBR_adm/GBR_adm2.shp")
uk_map <- fortify(uk_shp)

ggplot(uk_map, aes(x = long, y = lat, group=group)) + geom_path()
```

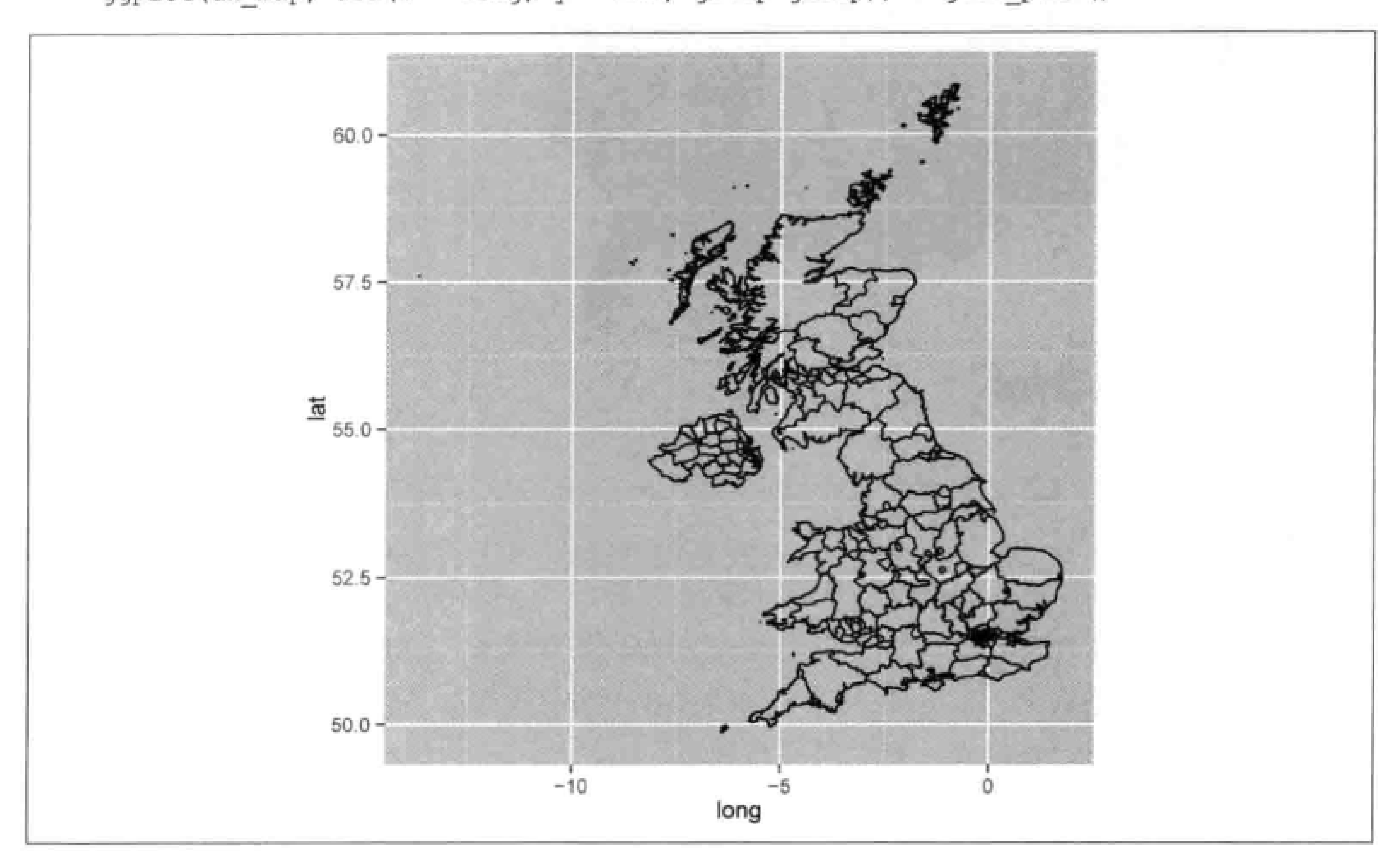


图 13-39　从 shapefile 创建的地图

讨论

Esri shapefile 是一种很常用的地图数据，`readShapePoly()` 函数读入空间数据文件，返回一个 `SpatialPolygonsDataFrame` 对象：

```
uk_shp <- readShapePoly("GBR_adm/GBR_adm2.shp")

# 查看该对象的格式
str(uk_shp)

Formal class 'SpatialPolygonsDataFrame' [package "sp"] with 5 slots
  ..@ data       :'data.frame': 192 obs. of  11 variables:
  .. ..$ ID_0     : int [1:192] 239 239 239 239 239 239 239 239 239 239 ...
```

```
  .. ..$ ISO      : Factor w/ 1 level "GBR": 1 1 1 1 1 1 1 1 1 1 ...
  .. ..$ NAME_0   : Factor w/ 1 level "United Kingdom": 1 1 1 1 1 1 1 1 1 1 ...
  .. ..$ ID_1     : int [1:192] 2 2 2 2 2 2 2 2 2 2 ...
 ... [lots more stuff]
  ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slots
  .. .. ..@ projargs: chr NA
```

把它转化为常规的数据框：

```
uk_map <- fortify(uk_shp)
uk_map

     long      lat order  hole piece group id
-6.210473 54.43324     1 FALSE     1   0.1  0
-6.166388 54.44416     2 FALSE     1   0.1  0
-6.165556 54.43417     3 FALSE     1   0.1  0
...
-3.053567 53.10680    57 FALSE     1 191.1 191
-2.991725 53.13901    58 FALSE     1 191.1 191
-2.956809 53.14467    59 FALSE     1 191.1 191
```

实际上，也可以直接在 ggplot() 中输入 SpatialPolygonsDataFrame 对象，这里自动使用了 fortify()：

```
# 直接在 ggplot() 中输入 SpatialPolygonsDataFrame
ggplot(uk_shp, aes(x=long, y=lat, group=group)) + geom_path()
```

尽管这个代码更加简单，但是你可能还是想自己用 fortify() 转化数据。这会使你更容易查看输入 ggplot() 中的数据的结构，或者把该数据框和其他数据集合并。

另见

这里使用的空间数据没有包含在 gcookbook 包中。该数据集和其他空间数据集都可以在这里下载得到：http://www.gadm.org/。

第14章

输出图形用以展示

一般而言，数据可视化为两个目标服务：发现和沟通。在发现阶段，你会创建探索性的图形，而在这样做的时候，能够快速尝试不同的可视化方式是比较重要的。在沟通阶段，你会把你的图形呈现给他人。当你做这件事的时候，往往需要微调图形的外观（在前面的章节已有叙述），并且经常需要将这些图形放置到你的计算机屏幕以外的地方。本章就是关于后一部分的：保存你的图形以使它们能够被呈现在文档中。

14.1 输出为 PDF 矢量文件

问题

如何为你的图形创建 PDF 格式的输出？

方法

有两种方法输出 PDF 文件。一种方法是，使用 pdf() 打开 PDF 图形设备，绘制图形，然后使用 dev.off() 关闭图形设备。这种方法适用于 R 中的大多数图形，包括基础图形和基于网格的图形，如那些由 ggplot2 和 lattice 创建的图形：

```
# width（宽度）和 height（高度）的单位为英寸
pdf("myplot.pdf", width=4, height=4)

# 绘制图形
plot(mtcars$wt, mtcars$mpg)
print(ggplot(mtcars, aes(x=wt, y=mpg)) + geom_point())

dev.off()
```

如果你绘制的图形多于一幅，则每一幅将在 PDF 输出中列于独立的一页。注意，我们针对 ggplot 对象调用了 print()，以确保这段代码即使是在一段脚本中也能够输出图形。

width（宽度）和 height（高度）的单位为英寸，所以，要以厘米为单位指定长宽，必须手动执行转换：

```
# 8×8 cm
pdf("myplot.pdf", width=8/2.54, height=8/2.54)
```

如果你使用某个脚本来创建图形，而在创建图形的过程中抛出了一个错误，则 R 可能无法执行到 dev.off() 这一步调用，并可能停留在 PDF 设备仍然开启的状态。当这种情况发生时，直到你去手动调用 dev.off() 之前，PDF 文件将无法正常打开：

如果你使用 ggplot2 创建图形，那么使用 ggsave() 会简单一些。此函数可以简单地保存使用 ggplot() 创建的最后一幅图形：

```
ggplot(mtcars, aes(x=wt, y=mpg)) + geom_point()

# 默认单位为英寸，不过也可指定单位
ggsave("myplot.pdf", width=8, height=8, units="cm")
```

使用 ggsave() 时，就无需打印 ggplot 对象了，并且如果在创建或保存图形时出现了错误，也无需手动关闭图形设备。不过，ggsave() 不能用于创建多页的图形。

讨论

当你的目的是输出到供打印的文档时，PDF 文件通常是最佳选择。这种格式可以与 LaTeX 轻松配合，而且可以用于苹果 Keynote 软件创建的演示幻灯片中，但微软的软件可能难以导入这种格式的文件（参见 14.3 节以了解关于如何创建可被导入微软软件的矢量图形的细节）。

PDF 文件一般也比位图文件，如便携式网络图形（PNG）文件更小，因为 PDF 文件包含的是一系列的指令，如“从这里到那里画一条直线”，而不是关于每个像素的颜色信息。不过，也存在位图文件更小的情况。例如，如果你有一幅存在严重遮盖的散点图，则 PDF 文件可能会比 PNG 文件大得多——虽然大多数点都无法彼此分清，PDF 文件却将仍然包含绘制每一个点的指令，反之，位图文件将不会包含这些冗余的信息。参见 5.5 节中的示例。

另见

如果你希望手动编辑 PDF 或 SVG 文件，参见 14.4 节。

14.2 输出为 SVG 矢量文件

问题

如何为你的图形创建可缩放矢量图形（SVG）格式的输出？

方法

SVG 文件在创建和使用的方法上与 PDF 文件基本相同：

```
svg("myplot.svg", width=4, height=4)
plot(...)
dev.off()

# 使用 ggsave()
ggsave("myplot.svg", width=8, height=8, units="cm")
```

讨论

当涉及导入图像时，某些程序可能在处理 SVG 方面比 PDF 要好些，反之亦然。举例来说，Web 浏览器一般有着更好的 SVG 支持，而文档创建程序，如 LaTeX，则一般有着更好的 PDF 支持。

14.3 输出为 WMF 矢量文件

问题

如何为你的图形创建 Windows 图元文件（WMF）格式的输出？

方法

WMF 文件在创建和使用的方法上与 PDF 文件基本相同——但这种格式的图形文件只能在 Windows 上创建[①]：

```
win.metafile("myplot.wmf", width=4, height=4)
plot(...)
dev.off()

# 使用 ggsave()
ggsave("myplot.wmf", width=8, height=8, units="cm")
```

讨论

Windows 下的程序，如 Microsoft Word 和 PowerPoint 对于 PDF 文件的导入支持较差，但是这些程序都是原生支持 WMF 格式的。WMF 格式的一个缺点是不支持透明（alpha 通道）。

14.4 编辑矢量格式的输出文件

问题

如何打开矢量格式的输出文件以用于最终的编辑？

---

① 使用 CRAN 上 devEMF 包提供的 emf() 设备，读者也可在 Linux 或 OSX 下创建 Windows 图元格式的文件。关于这种格式的更多评论，可以参见 http://cos.name/cn/topic/109475。——译者注

方法

有时为了展示图形，我们需要对一幅图的外观进行最终的微调。你可以使用出色的自由软件 Inkscape 或商业软件 Adobe Illustrator 打开 PDF 和 SVG 文件。

讨论

当你使用 Inkscape 打开 PDF 文件时，字体支持可能是一个问题。一般来说，使用 PDF 设备绘制的点对象将被编码为 Zapf Dingbats 字体中的符号。这样在你希望使用像 Illustrator 或 Inkscape 一类的编辑器打开这些文件时可能会有问题；举例来说，点可能会显示为字母 q，如图 14-1 所示，因为这正是实心圆形项目符号在 Zapf Dingbats 中对应的字母。

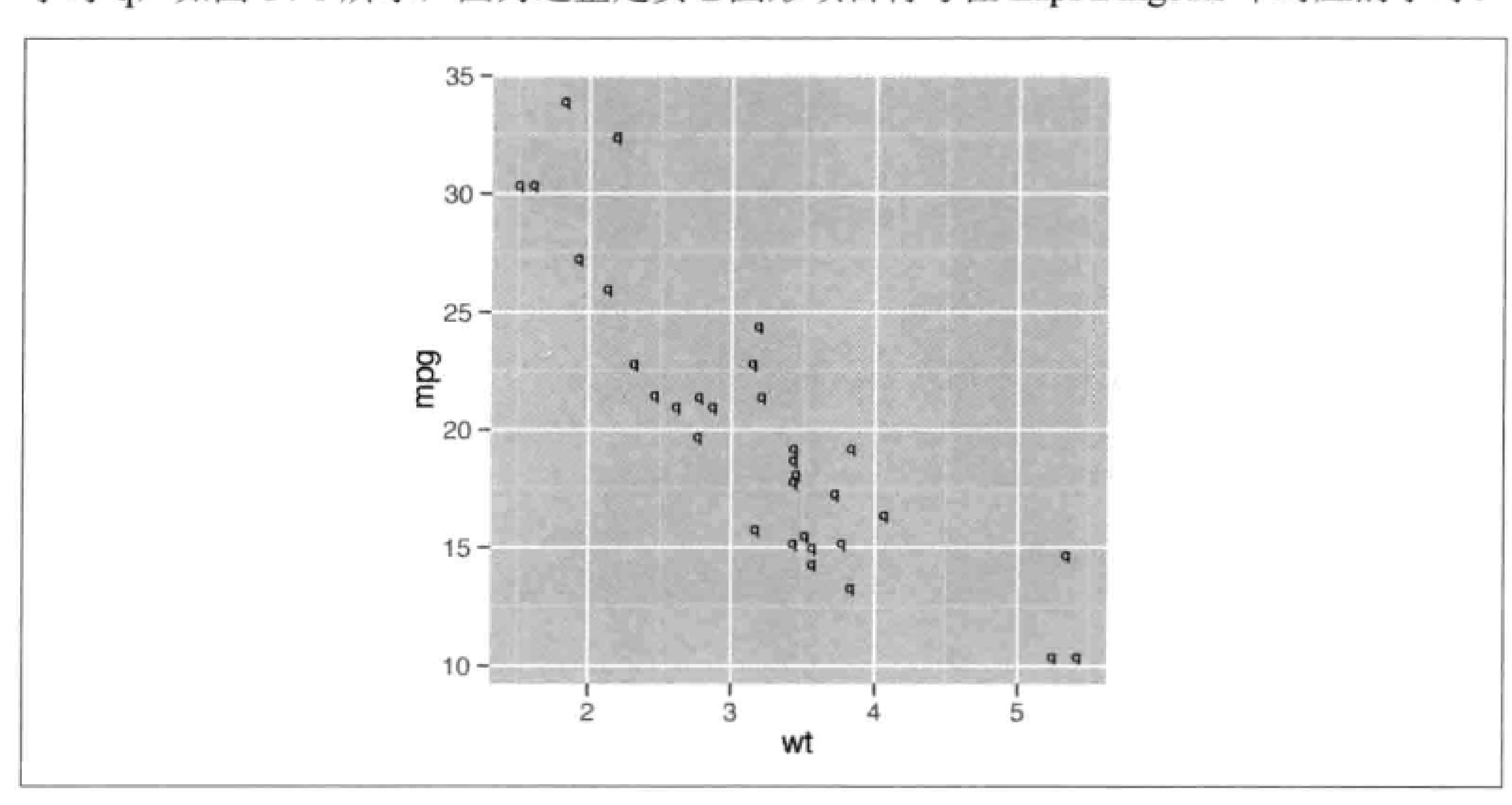
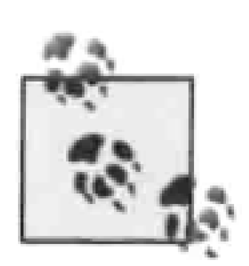


图 14-1　在 Inkscape 中打开后对点符号的糟糕转换——同时注意到文字的间距也略有问题

要避免这个问题，设置参数 useDingbats=FALSE 即可。这将使得圆圈被绘制为圆圈而不是字体中的字符：

```
pdf("myplot.pdf", width=4, height=4, useDingbats=FALSE)

# 或者
ggsave("myplot.pdf", width=4, height=4, useDingbats=FALSE)
```



即使进行如上操作，Inkscape 在字体方面可能依然存在一些问题。你可能已经注意到了图 14-1 中的字体看起来不太对。这是因为 Inkscape（0.48 版本）无法找到 Helvetica 字体，并使用 Bitstream Vera Sans 字体替代了前者。一种临时解决方案是复制 Helvetica 字体文件到你的字体库中。举例来说，在 Mac OS X 上，在终端窗口中执行 `cp /System/Library/Fonts/Helvetica.dfont ~/Library/Fonts/` 即可，当提示说有字体冲突时，点击“忽略冲突”。在完成以后，Inkscape 就应该可以正常地显示 Helvetica 字体了。

14.5 输出为点阵（PNG/TIFF）文件

问题

如何创建点阵格式的图形，并写入到 PNG 文件中？

方法

有两种方法可以输出 PNG 点阵文件。一种方法是使用 png() 打开 PNG 图形设备，绘制图形，然后使用 dev.off() 关闭设备。这种方法对于 R 中的多数图形都有效，包括基础图形和基于网格的图形，如那些由 ggplot2 和 lattice 创建的图形：

```
# width（宽度）和 height（高度）的单位为像素
png("myplot.png", width=400, height=400)

# 绘制图形
plot(mtcars$wt, mtcars$mpg)

dev.off()
```

要输出多幅图形，可在文件名中加入 %d。对于后续图形，这个位置将被 1、2、3 等替代：

```
# width（宽度）和 height（高度）的单位为像素
png("myplot-%d.png", width=400, height=400)

plot(mtcars$wt, mtcars$mpg)
print(ggplot(mtcars, aes(x=wt, y=mpg)) + geom_point())

dev.off()
```

注意，我们针对 ggplot 对象调用了 print()，以确保这段代码即使是在一段脚本中时也能够输出图形。

width（宽度）和 height（高度）的单位为像素，而默认的输出分辨率为每英寸 72 像素（72 ppi）。这一分辨率适合在屏幕上显示，但会在打印时显得模糊且有锯齿。

对于高质量的打印输出，分辨率应至少为 300 ppi。图 14-2 显示了相同图形的一部分在不同分辨率下的效果。在本例中，我们将使用 300 ppi 的设置，并创建一个 4 英寸 ×4 英寸的 PNG 文件：

```
ppi <- 300
# 计算一幅 4 英寸 ×4 英寸 300 ppi 图像的高度和宽度（以像素为单位）

png("myplot.png", width=4*ppi, height=4*ppi, res=ppi)
plot(mtcars$wt, mtcars$mpg)
dev.off()
```

如果你使用某个脚本来创建图形，而在创建图形的过程中抛出了一个错误，则 R 可能无法执行到 dev.off() 这一步调用，并可能停留在 PNG 设备仍然开启的状态。当这种情况发生时，直到你去手动调用 dev.off() 之前，PNG 文件将无法用查看程序正常地打开：

如果你使用 ggplot2 创建图形，那么使用 ggsave() 会简单一些。此函数可以简单地保存使用 ggplot() 创建的最后一幅图形。你可以以英寸而不是以像素为单位指定图形的宽度和高度，并指定使用的 ppi 数值：

```
ggplot(mtcars, aes(x=wt, y=mpg)) + geom_point()

# 默认的宽高单位是英寸，不过也可以指定其他单位
ggsave("myplot.png", width=8, height=8, unit="cm", dpi=300)
```

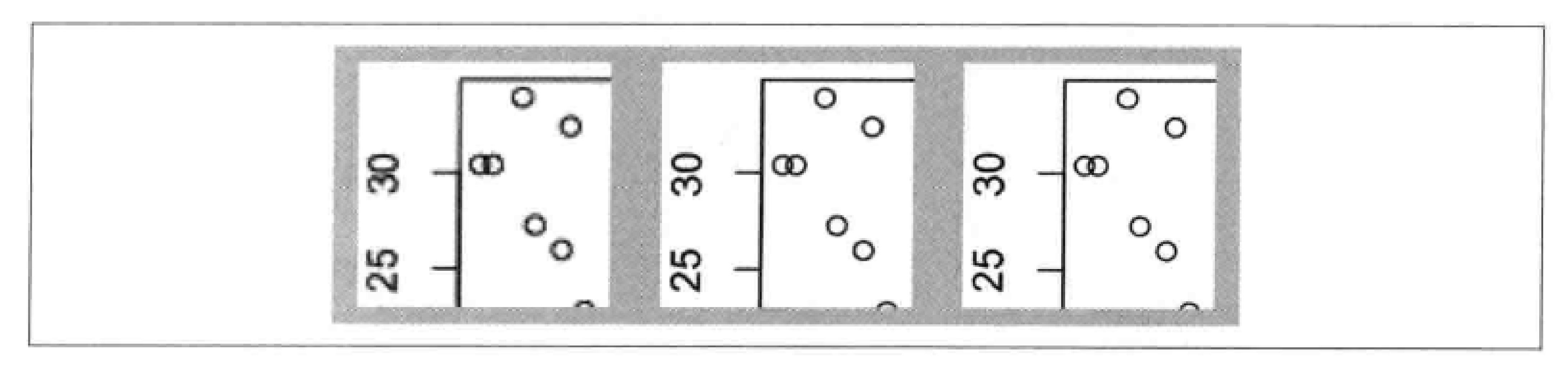


图 14-2　从左至右：72、150 和 300 ppi 的 PNG 输出（实际大小）

使用 ggsave() 时，无需打印 ggplot 对象，并且如果在创建或保存图形时出现了错误，也无需手动关闭图形设备。



虽然这个参数名为 dpi，但它实际上控制的却是每英寸的像素数（pixels per inch，ppi），而非每英寸的点数（dots per inch，dpi）。在印刷中渲染一个灰色像素时，我们是使用许多黑色墨水的小点来输出它的——因此印刷输出拥有的每英寸点数要高于每英寸像素数。

讨论

R 也支持其他的点阵图形格式，如 BMP、TIFF 和 JPEG，但其实真的没有太多理由去使用这些格式而不去使用 PNG 格式。

点阵图形具体外观视平台而异。与 R 中的 PDF 输出设备跨平台渲染一致的特点不同的是，点阵输出设备可能在 Windows、Linux 和 Mac OS X 上对相同图形渲染出不同的效果。甚至在相同类型的操作系统中也可能会有差异。

不同的平台对字体的渲染不同，某些平台会对线条进行抗锯齿（平滑）处理，而其他平台可能不会。某些平台支持 alpha 通道（透明度），而其他平台可能并不支持。如果你的平台缺乏对像抗锯齿和 alpha 通道之类特性的支持，可以使用 Cairo 包中的 CairoPNG() 设备：

```
install.packages("Cairo")  # 只需安装一次
CairoPNG("myplot.png")
plot(...)
dev.off()
```

尽管 CairoPNG() 并不能保证跨平台精确一致的渲染（字体可能不会完全相同），它还是支持如抗锯齿和 alpha 通道这样的特性。

修改分辨率会影响图形对象，如文本、线条和点这类图形对象的大小（以像素为单位）。举例来说，一幅 6 英寸 ×6 英寸大小，75 ppi 的图像在以像素衡量的宽高与一幅 3 英寸 ×3 英寸，150 ppi 的图像是相同的，但外观是不同的，如图 14-3 所示。这些图像均为 450 像素 ×450 像素。和这里一样，当在计算机屏幕上显示时，它们大小几乎相同。

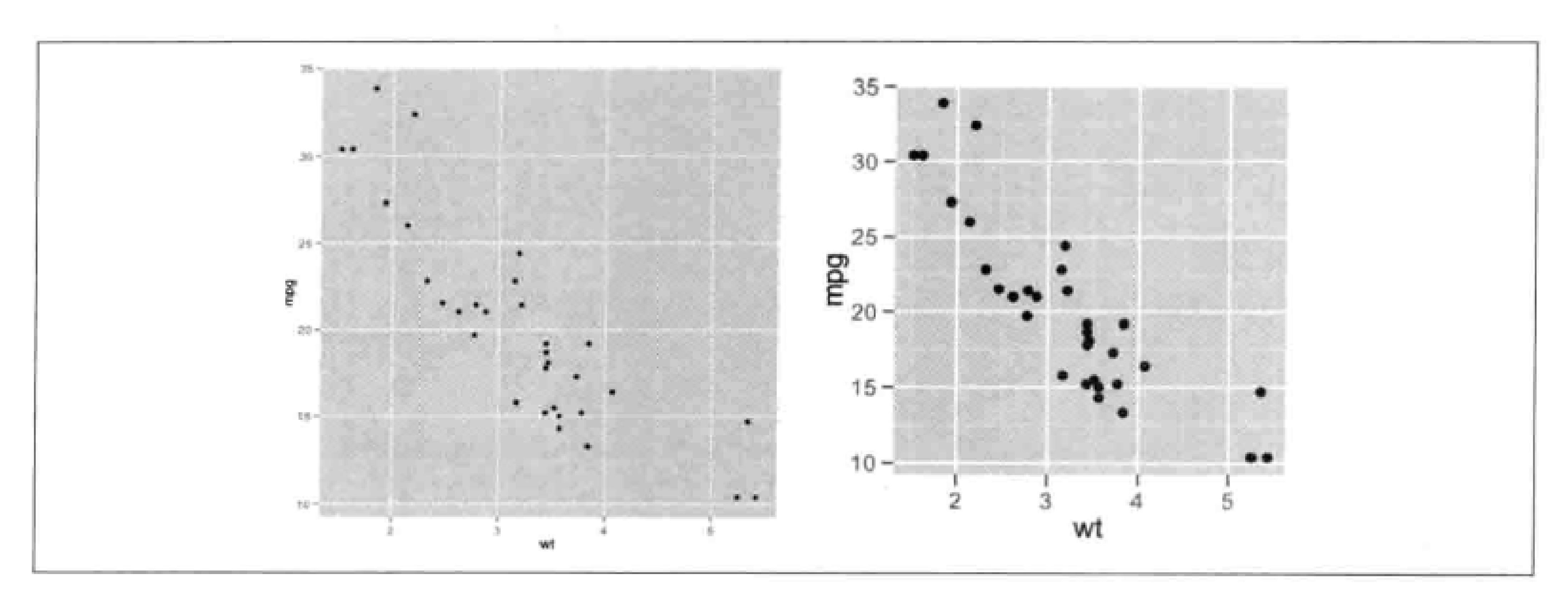


图 14-3　左图：6 英寸 × 6 英寸 75 ppi 的图像　右图：3 英寸 × 3 英寸 150 ppi 的图像

14.6　在 PDF 文件中使用字体

问题

如何在 PDF 文件中使用由 R 提供的基本字体以外的字体？

方法[1]

extrafont 包可用于创建包含其他字体的 PDF 文件。

这个过程涉及许多步骤，首先是一些一次性的软件安装和配置。下载并安装 Ghostscript，然后在 R 中执行以下命令：

```
install.packages("extrafont")
library(extrafont)

# 查找并保存系统中已安装字体的信息
font_import()

# 列出字体
fonts()
```

在一次性的安装和设置完成后，需要你在每个新的 R 会话中执行的是：

① 目录更好的解决方法是使用邱怡轩开发的 showtext 包，该包可以非常简单、灵活地在各种图形设备中添加字体，具体参见 COS 主站文章《showtext：字体，好玩的字体和好玩的图》，地址：http://cos.name/2014/01/showtext-interesting-fonts-and-graphs/。——译者注

```
library(extrafont)
# 在R中注册字体
loadfonts()

# 在 Windows 上，你可能需要指定 Ghostscript 的安装位置
#（根据你的 Ghostscript 安装位置调整对应的路径）
Sys.setenv(R_GSCMD = "C:/Program Files/gs/gs9.05/bin/gswin32c.exe")
```

最后，你可以创建 PDF 文件并向其中嵌入字体，如图 14-4 所示：

```
library(ggplot2)
ggplot(mtcars, aes(x=wt, y=mpg)) + geom_point() +
    ggtitle("Title text goes here") +
    theme(text = element_text(size = 16, family="Impact"))

ggsave("myplot.pdf", width=4, height=4)

embed_fonts("myplot.pdf")
```

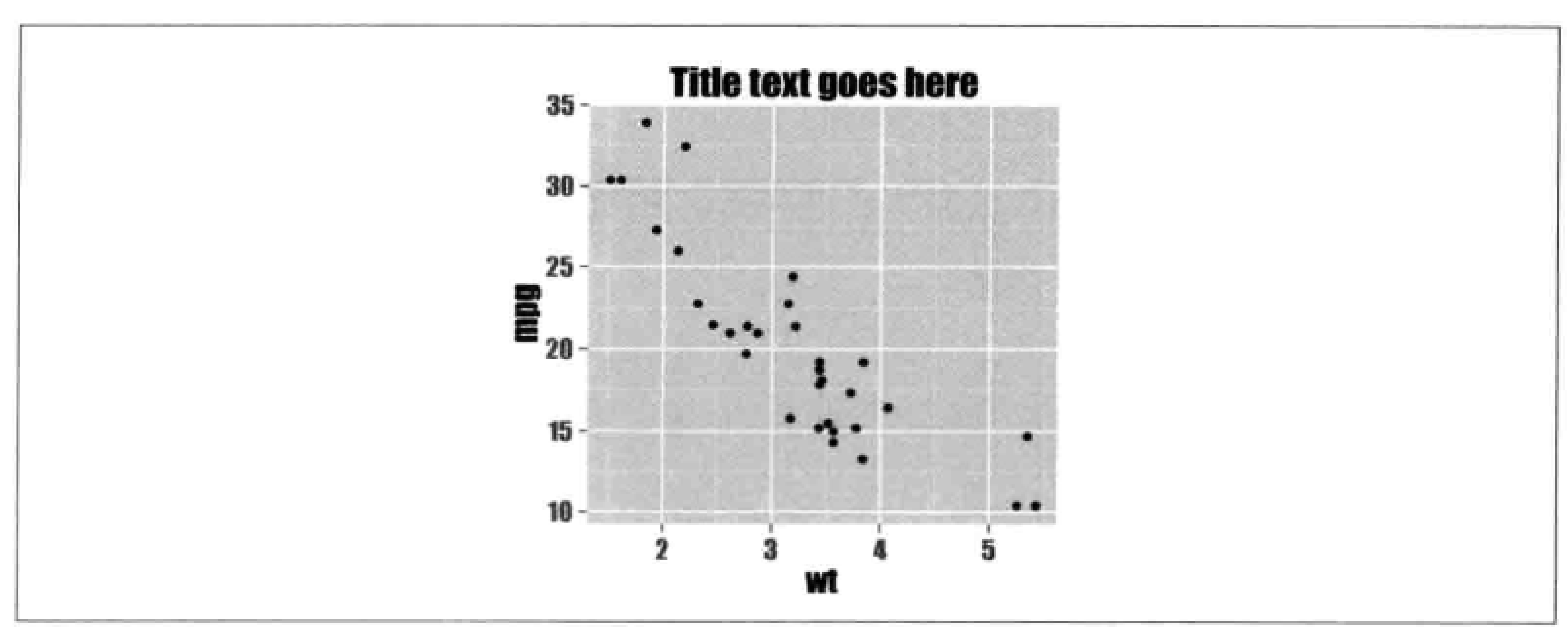


图 14-4　嵌入了 Impact 字体的 PDF 输出

讨论

在 R 中使用字体可能会很困难。某些输出设备，如 Mac OS X 中的屏幕显示设备 quartz 可以显示计算机上已安装的任意字体。其他输出设备，如 Windows 上的默认 png 设备，却无法显示系统字体。

此外，PDF 文件在涉及字体的方面也有自己的怪异之处。PDF 规范中指定了 14 种“核心”字体。这些字体是每个 PDF 渲染器都拥有的，其中包括标准字体，如 Times、Helvetica 和 Courier。如果你创建的 PDF 中使用了这些字体，则任何 PDF 渲染器都应该可以正确地显示它。

但是如果你希望使用除了这些核心字体以外的某种字体，则无法保证某个设备上的 PDF 渲染器拥有这种字体，所以你无法确认这种字体是否能够在其他计算机或打印机

上正确地显示或打印。为了解决这个问题，非核心字体可以被嵌入到 PDF 中；换句话说，PDF 文件本身可以包含一份你希望使用字体的副本。

如果你在一个 PDF 文档中放置了多幅 PDF 图形，则可能会希望在最终的文档中嵌入相应的字体而不是在每幅图形中都嵌入一次。这样可以使得最终的文档小一些，因为字体只被嵌入了一次，而不是每幅图形都嵌入一次。

使用 R 来嵌入字体可能是一个棘手的过程，不过 extrafont 包已经替你处理了许多讨厌的细节。



在写作本段之时，extrafont 仅可导入 TrueType（*.ttf*）字体，但它可能会在将来支持其他常见的字体格式，如 OpenType（*.otf*）字体。

另见

关于控制文本外观的更多内容，参见 9.2 节。

14.7 在 Windows 的点阵或屏幕输出中使用字体

问题

使用 Windows 时，如何在点阵或屏幕输出中使用除了 R 中提供的基本字体以外的字体？

方法

extrafont 包可以用于创建点阵或屏幕输出。过程与使用 extrafont 处理 PDF 文件类似（参见 14.6 节）。除了不需要 Ghostscript 以外，一次性的安装过程几乎是相同的：

```
install.packages("extrafont")
library(extrafont)

# 查找并保存系统中已安装字体的信息
font_import()

# 列出字体
fonts()
```

在一次性的安装过程结束之后，需要你在每个新的 R 会话中执行的是：

```
library(extrafont)
# 注册 Windows 中的字体
loadfonts("win")
```

最后，你可以创建输出文件或在屏幕上显示图形，如图 14-5 所示：

```
library(ggplot2)
ggplot(mtcars, aes(x=wt, y=mpg)) + geom_point() +
    ggtitle("Title text goes here") +
    theme(text = element_text(size = 16, family="Georgia", face="italic"))

ggsave("myplot.png", width=4, height=4, dpi=300)
```

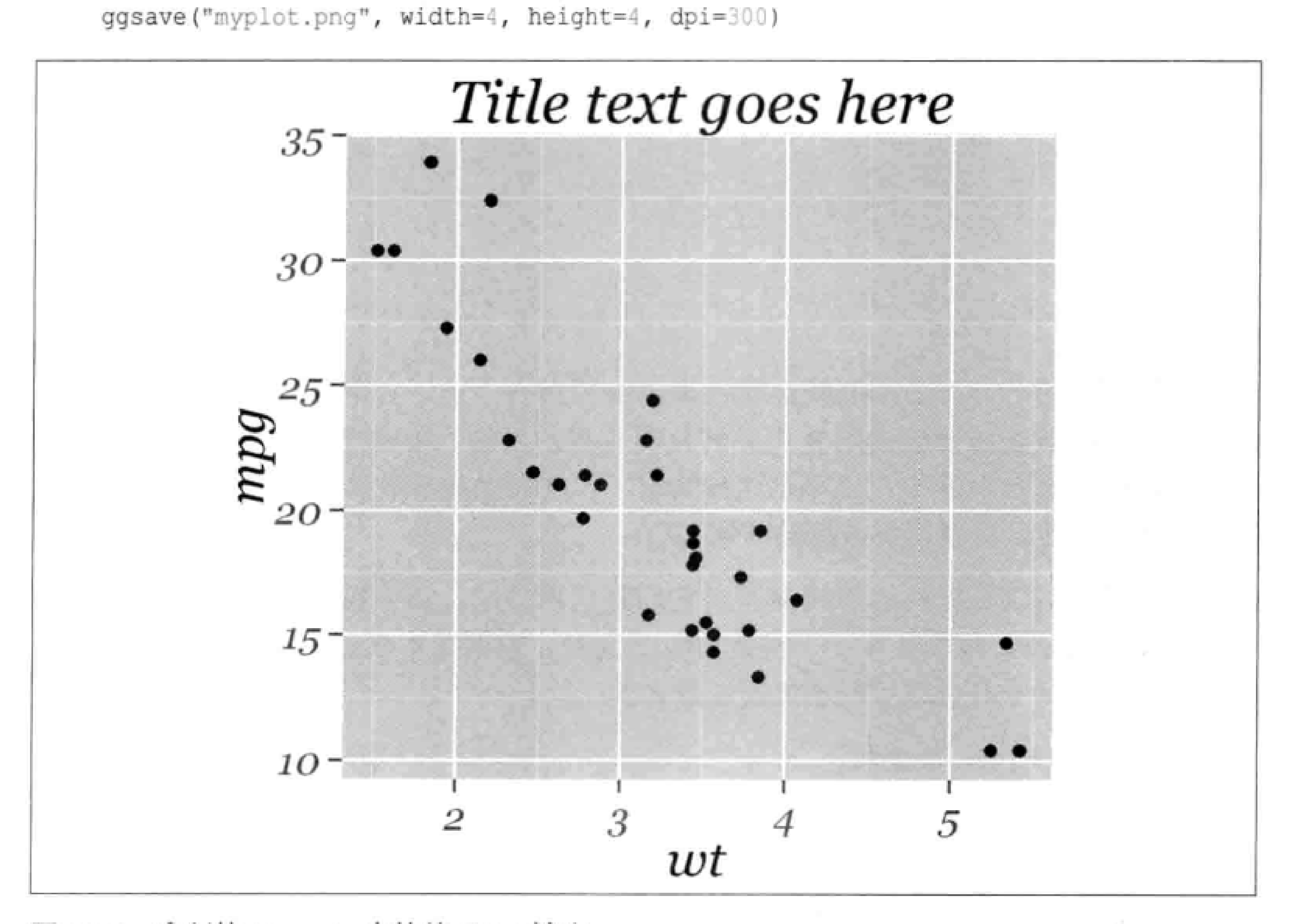


图 14-5　含斜体 Georgia 字体的 PNG 输出

讨论

点阵图形与 PDF 图形在字体的处理方式上是完全不同的。

在 Windows 上，对于点阵输出，必须手工在 R 中注册每一个字体（extrafont 让这件事变得容易了许多）。而在 Mac OS X 和 Linux 上，对于点阵输出，字体应该是直接可用的，并不需要手工注册它们。

第15章

数据塑形

绘制统计图形时，有半数的时间会花在调用绘图命令之前。在你把数据传送给画图函数之前，数据首先需要被读入 R 中并转化为正确的结构。R 中提供的数据集可以直接使用，但到了真实世界中，就不是这种情况了：在将数据转化为图形之前，你需要对数据进行清理然后重新组织数据的结构。

R 中的数据集常以数据框的形式存在。它们都是典型的二维数据结构，每行代表一个具体对象（case），每列代表一个描述对象的变量。数据框本质上是由向量和因子组成的列表，其中每个向量或者因子代表了数据的一列。

下面是 heightweight 数据集：

```
library(gcookbook) # 为了使用数据集
heightweight

sex ageYear ageMonth heightIn weightLb
  f   11.92      143     56.3     85.0
  f   12.92      155     62.3    105.0
 ...
  m   13.92      167     62.0    107.5
  m   12.58      151     59.3     87.0
```

它一共五列，每行代表了一个具体对象（case）：某个人的一些信息。我们可以通过 str() 函数清楚地了解它的结构。

```
str(heightweight)

'data.frame': 236 obs. of 5 variables:
$ sex      : Factor w/ 2 levels "f","m": 1 1 1 1 1 1 1 1 1 1 ...
$ ageYear : num 11.9 12.9 12.8 13.4 15.9 ...
$ ageMonth: int 143 155 153 161 191 171 185 142 160 140 ...
$ heightIn: num 56.3 62.3 63.3 59 62.5 62.5 59 56.5 62 53.8 ...
$ weightLb: num 85 105 108 92 112 ...
```

数据的第一列 sex，是一个两水平（"f" 和 "m"）的因子。其余四列都是数值型向量（其中的 ageMonth 是整型向量，但在此处，它与其他数值向量并无明显区别）。

因子和字符型向量在 ggplot2 中的处理方式相类似——主要区别在于，展示字符型向量代表的项目时，它们以字母表的顺序排列，而因子型的项目是按因子水平的顺序排列，这个顺序是可以由用户控制的。

15.1 创建数据框

问题

如何将若干向量组织成数据框？

方法

你可以把向量放在 data.frame() 里面：

```
# 从两个简单的向量开始
g <- c("A", "B", "C")
x <- 1:3

dat <- data.frame(g, x)
dat

 g x
 A 1
 B 2
 C 3
```

讨论

数据框本质上是由一堆向量和因子构成的列表，其中每个向量或者因子代表了一列。

如果你的向量在一个列表中，你可以用 as.data.frame() 函数直接将它们转化成数据框：

```
lst <- list(group = g, value = x) # 由向量组成的列表

dat <- as.data.frame(lst)
```

15.2 从数据框中提取信息

问题

如何从一个对象或者数据框中提取主要信息？

方法

使用 str() 函数：

```
str(ToothGrowth)
```

```
'data.frame': 60 obs. of 3 variables:
 $ len : num 4.2 11.5 7.3 5.8 6.4 10 11.2 11.2 5.2 7 ...
 $ supp: Factor w/ 2 levels "OJ","VC": 2 2 2 2 2 2 2 2 2 2 ...
 $ dose: num 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 ...
```

以上代码告诉你 ToothGrowth 由 len、supp 和 dose 三列组成，len 和 dose 包含的是数值变量，而 supp 是一个有两个水平的因子。

讨论

str() 函数在提取数据框更多信息的时候很实用。一个常见的问题是，有时候一个数据框包含的向量是字符型向量而不是因子，反之亦然。这个问题会在分析和画图的时候给我们造成一些困扰。

当你想用常规的方式输出一个数据框时，只需要在提示符（>）后面输入数据框的名称，然后敲下回车键。你会发现字符型向量和因子的输出效果一样，你无法分辨出哪个是字符型向量，哪个是因子。只有当你运行 str() 函数或者单独输出一列的时候，才能看出它们的区别：

```
tg <- ToothGrowth
tg$supp <- as.character(tg$supp)

str(tg)

'data.frame': 60 obs. of 3 variables:
 $ len : num 4.2 11.5 7.3 5.8 6.4 10 11.2 11.2 5.2 7 ...
 $ supp: chr "VC" "VC" "VC" "VC" ...
 $ dose: num 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 ...

# 直接输出列

# 原始数据框（因子）
ToothGrowth$supp

 [1] VC VC VC VC VC VC VC VC VC VC VC VC VC VC VC VC VC VC VC VC VC VC VC VC VC
[26] VC VC VC VC VC OJ OJ OJ OJ OJ OJ OJ OJ OJ OJ OJ OJ OJ OJ OJ OJ OJ OJ OJ OJ
[51] OJ OJ OJ OJ OJ OJ OJ OJ OJ OJ
Levels: OJ VC

# 新数据框（字符串）
tg$supp

 [1] "VC" "VC" "VC" "VC" "VC" "VC" "VC" "VC" "VC" "VC" "VC" "VC" "VC" "VC" "VC"
[16] "VC" "VC" "VC" "VC" "VC" "VC" "VC" "VC" "VC" "VC" "VC" "VC" "VC" "VC" "VC"
[31] "OJ" "OJ" "OJ" "OJ" "OJ" "OJ" "OJ" "OJ" "OJ" "OJ" "OJ" "OJ" "OJ" "OJ" "OJ"
[46] "OJ" "OJ" "OJ" "OJ" "OJ" "OJ" "OJ" "OJ" "OJ" "OJ" "OJ" "OJ" "OJ" "OJ" "OJ"
```

15.3 向数据框添加列

问题

如何在数据框中添加列？

方法

只需把值赋到新的列即可。

如果你把单个值赋到一个新的列，那么整个列都会被赋予这个值。下面的例子是增加一个新的列，值全部是 NA：

```
data$newcol <- NA
```

你也可以把一个向量赋到新的一列：

```
data$newcol <- vec
```

如果该向量的长度比数据框的行数小，那么R会重复这个向量，直到所有的行被填充。

讨论

数据框的每一列都是一个向量或者因子。R 在处理数据框的时候会和处理单独的向量时略有不同，因为在数据框中所有列的长度都是一样的。

15.4　从数据框中删除一列

问题

如何从数据框中删除一列？

方法

把该列的值赋成 NULL 即可。

```
data$badcol <- NULL
```

讨论

你也可以使用 subset() 函数并将一个 -（减号）置于待删除的列之前：

```
# 返回不包含 badcol 列的数据
data <- subset(data, select = -badcol)

# 排除 badcol 列和 othercol 列
data <- subset(data, select = c(-badcol, -othercol))
```

另见

更多关于获取数据框子集的信息，参见 15.7 节。

15.5　重命名数据框的列名

问题

如何重命名数据框的列名？

方法

使用 names(dat) <- 函数即可：

```
names(dat) <- c("name1", "name2", "name3")
```

讨论

如果你想通过列名重命名某一列：

```
library(gcookbook) # 为了使用数据集
names(anthoming)   # 输出列名

 "angle" "expt" "ctrl"

names(anthoming)[names(anthoming) == "ctrl"] <- c("Control")
names(anthoming)[names(anthoming) == "expt"] <- c("Experimental")
names(anthoming)

 "angle"   "Experimental" "Control"
```

你也可以通过名字的数值位置重命名：

```
names(anthoming)[1] <- "Angle"
names(anthoming)

 "Angle"   "Experimental" "Control"
```

15.6 重排序数据框的列

问题

如何改变数据框中列的顺序？

方法

通过列的数值位置重排序：

```
dat <- dat[c(1,3,2)]
```

通过列的名称重排序：

```
dat <- dat[c("col1", "col3", "col2")]
```

讨论

前面的例子用了一个列表形式的索引。数据框本质上是若干向量组成的列表，如果改变其索引，我们会得到另一个数据框。我们用矩阵形式的索引也能得到同样的效果：

```
library(gcookbook) # 为了使用数据集
anthoming

 angle expt ctrl
   -20    1    0
   -10    7    3
```

```
     0     2     3
    10     0     3
    20     0     1
anthoming[c(1,3,2)] # 列表风格的索引

 angle ctrl expt
   -20    0    1
   -10    3    7
     0    3    2
    10    3    0
    20    1    0

# 逗号之前的空白表示输出所有行
anthoming[, c(1,3,2)] # 矩阵风格的索引

 angle ctrl expt
   -20    0    1
   -10    3    7
     0    3    2
    10    3    0
    20    1    0
```

这种情况下，两种方法都会得到数据框。然而，当你检索单独一列的时候，列表风格的索引会得到数据框，而矩阵风格的索引得到的是向量，除非你加上参数 drop=FALSE：

```
anthoming[3] # 列表风格的索引

 ctrl
    0
    3
    3
    3
    1
anthoming[, 3] # 矩阵风格的索引

 0 3 3 3 1

anthoming[, 3, drop=FALSE] # 矩阵风格的索引，并添加参数 drop=FALSE

 ctrl
    0
    3
    3
    3
    1
```

15.7 从数据框提取子集

问题

如何得到数据框的子集？

方法

使用 subset() 函数。它可以筛选出符合一系列条件的行和选出特定的列。

我们用 climate 数据集作为例子：

```
library(gcookbook) # 为了使用数据集
climate

  Source Year Anomaly1y Anomaly5y Anomaly10y Unc10y
Berkeley 1800        NA        NA     -0.435  0.505
Berkeley 1801        NA        NA     -0.453  0.493
Berkeley 1802        NA        NA     -0.460  0.486
 ...
 CRUTEM3 2009    0.7343        NA         NA     NA
 CRUTEM3 2010    0.8023        NA         NA     NA
 CRUTEM3 2011    0.6193        NA         NA     NA
```

下面的代码只会输出 Source 是 "Berkeley" 的行，并且选取名字为 Year 和 Anomaly10y 的列：

```
subset(climate, Source == "Berkeley", select = c(Year, Anomaly10y))

Year Anomaly10y
1800     -0.435
1801     -0.453
1802     -0.460
 ...
2002      0.856
2003      0.869
2004      0.884
```

讨论

我们还可以通过使用 |（OR）和 &（AND）操作符同时施加多种筛选条件。例如，下面的代码会筛选出 Source 是 "Berkeley" 并且 Year 在 1900 和 2000 之间的数据：

```
subset(climate, Source == "Berkeley" & Year >= 1900 & Year <= 2000,
       select = c(Year, Anomaly10y))

Year Anomaly10y
1900     -0.171
1901     -0.162
1902     -0.177
 ...
1998      0.680
1999      0.734
2000      0.748
```

你也可以在方括号里面加入索引来得到子数据框，虽然这种方法不是很优雅。下面的代码和上面用到的代码有同样的效果。方括号里面、逗号前面的部分提取行，逗号后

面的部分提取列：

```
climate[climate$Source=="Berkeley" & climate$Year >= 1900 & climate$Year <= 2000,
        c("Year", "Anomaly10y")]
```

如果用这种方式得到的结果只有一列，那么它会返回一个向量而不是一个数据框，除非你使用参数 drop=FALSE：

```
climate[climate$Source=="Berkeley" & climate$Year >= 1900 & climate$Year <= 2000,
        c("Year", "Anomaly10y"), drop=FALSE]
```

最后，我们还可以通过行和列的数值位置提取子数据框。下面代码得到的是前 100 行的第二和第五列：

```
climate[1:100, c(2, 5)]
```

我建议尽可能地使用名称索引，避免使用数字索引。前者使代码更加易懂，尤其是你和其他人合作的时候或者在你完成代码几个月甚至几年以后回过头来再看的时候，也使得代码不容易因为数据框行和列的增减而得不到原先的效果。

15.8 改变因子水平的顺序

问题

如何改变因子水平的顺序？

方法

因子的水平可以由函数 factor() 具体设定。在下面这个例子中，我们创造了一个因子，但是它的因子水平的顺序是乱的：

```
# 默认的因子水平的顺序是按字母排的
sizes <- factor(c("small", "large", "large", "small", "medium"))
sizes

small large large small medium
Levels: large medium small

# 改变因子水平的顺序
sizes <- factor(sizes, levels = c("small", "medium", "large"))
sizes

small  large  large  small  medium
Levels: small medium large
```

因子的顺序也可以在第一次创建因子时通过 levels 参数来指定。

讨论

R 中有两种因子：顺序因子（orderer factor）和常规因子（regular factor）。在两种类型中，因子水平都是按某种顺序排列的；区别在于，对于顺序因子，因子水平的顺序是有意义的，

而对于常规因子，因子水平的顺序却没有什么意义——它仅仅是反映了数据是如何存储的。对于用于画图的数据，两者的区别一般来说不太重要，因为处理它们的方式是一样的。

因子水平的顺序会影响图形输出。当一个因子变量被映射到 ggplot2 中的图形属性中，图形属性会采用因子水平的顺序。如果因子被映射到 x 轴，x 轴的标签会按因子水平的顺序排列；如果因子被映射到颜色上，那么图例会按因子水平的顺序排序。

如果要颠倒因子水平的顺序，可以使用函数 rev(levels())：

```
factor(sizes, levels = rev(levels(sizes)))

small  large  large  small  medium
Levels: small medium large
```

另见

如果要依据其他变量的值对因子水平排序，参见 15.9 节。

改变因子的顺序对控制坐标标签和图例的顺序很有用。参见 8.3 节和 10.3 节。

15.9　根据数据的值改变因子水平的顺序

问题

如何根据数据的值改变因子水平的顺序？

方法

使用函数 reorder()，该函数有三个参数：因子，排序依据的数据和汇总数据的函数。

```
# 复制一份数据，因为我们要修改它
iss <- InsectSprays
iss$spray

 [1] A A A A A A A A A A A A B B B B B B B B B B B B C C C C C C C C C C C C D D
[39] D D D D D D D D D D E E E E E E E E E E E E F F F F F F F F F F F F
Levels: A B C D E F

iss$spray <- reorder(iss$spray, iss$count, FUN=mean)
iss$spray

 [1] A A A A A A A A A A A A B B B B B B B B B B B B C C C C C C C C C C C C D D
[39] D D D D D D D D D D E E E E E E E E E E E E F F F F F F F F F F F F
attr(,"scores")
        A         B         C         D         E         F
14.500000 15.333333  2.083333  4.916667  3.500000 16.666667
Levels: C E D A B F
```

原始因子水平的顺序是 ABCDEF，重排后的顺序是 CEDABF。新的顺序是由 iss$spray 中每组 iss$count 的平均值决定的。

讨论

仅从简单的输出我们无法看出 reorder() 的作用。图 15-1 给我们展示了经过 reorder() 函数排序后的三个图形。在这些图形中，每一项出现的顺序是由它们的某些值决定的。

图 15-1 中间的箱线图是按照每组的平均值排序的。箱线图中箱子里的水平线表示了这一组的中位数。注意到这些中位数并不是从左到右严格递增的。这是因为根据均值和根据中位数排序得到的顺序是不一样的。为了使中位数线从左到右递增，就像 15-1 右边的图一样，我们必须按每组的中位数排序，于是在 reorder() 中，我们应该使用函数 median()。

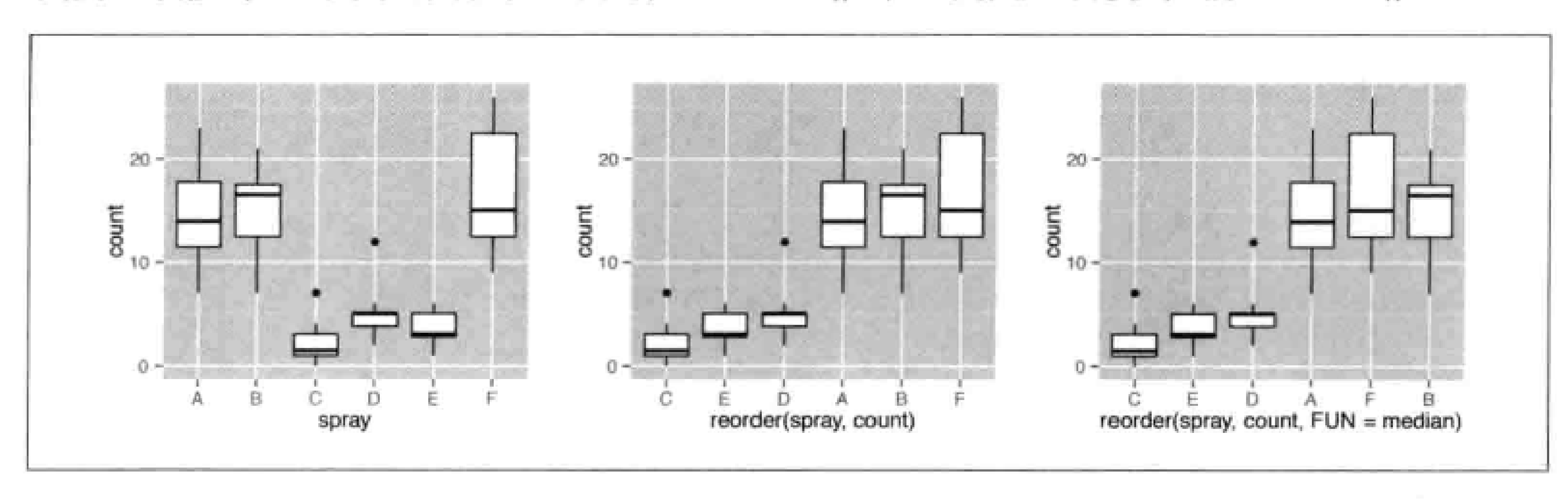


图 15-1　左图：原始数据　中图：以每组平均值排序后的数据　右图：以每组中位数排序后的数据

另见

改变因子的顺序对控制坐标标签的顺序和图例的顺序很有用，参见 8.3 节和 10.3 节。

15.10　改变因子水平的名称

问题

如何改变因子水平的名称？

方法

使用 plyr 包中的 revalue() 函数或 mapvalues() 函数。

```
sizes <- factor(c( "small", "large", "large", "small", "medium"))
sizes

small  large  large  small  medium
Levels: large medium small

levels(sizes)

"large" "medium" "small"

# 通过函数 revalue()，传递一组映射关系
sizes1 <- revalue(sizes, c(small="S", medium="M", large="L"))
sizes1
```

```
S L L S M
Levels: L M S
```

```
# 也可以使用引号 —— 如果原因子水平名称中存在空格等特殊字符，这将很有用
revalue(sizes, c("small"="S", "medium"="M", "large"="L"))

# mapvalues() 函数使用两组向量，而不是一组映射关系向量
mapvalues(sizes, c("small", "medium", "large"), c("S", "M", "L"))
```

讨论

`revalue()` 函数和 `mapvalues()` 函数很方便，但是在 R 中有一个更传统（也更笨重）的方法，使用 `levels()<-` 函数：

```
sizes <- factor(c( "small", "large", "large", "small", "medium"))

# 通过水平原名称找到某个水平然后重命名
levels(sizes)[levels(sizes)=="large"] <- "L"
levels(sizes)[levels(sizes)=="medium"] <- "M"
levels(sizes)[levels(sizes)=="small"] <- "S"
sizes
```

```
S L L S M
Levels: L M S
```

如果你要改变所有水平的名称，这里有一个更简单的方法。你可以给 `levels()` 传递一个 `list` 类型的参数：

```
sizes <- factor(c("small", "large", "large", "small", "medium"))
levels(sizes) <- list(S="small", M="medium", L="large")
sizes
```

```
S L L S M
Levels: L M S
```

在这个方法中，所有的因子水平必须在一个 `list` 里面指定；如果这个 `list` 里面有任何的缺失，缺失的值最终会以 `NA` 代替。

通过因子水平的位置也可以重命名，但是这种方法比较笨：

```
# 默认情况下，因子水平是按字母顺序排列的
sizes <- factor(c("small", "large", "large", "small", "medium"))
```

```
small  large  large  small  medium
Levels: large medium small
```

```
levels(sizes)[1] <- "L"
sizes
```

```
small  L      L      small medium
Levels: L medium small
```

```
# 一次重命名所有的水平
levels(sizes) <- c("L", "M", "S")
sizes
```

```
[1] S L L S M
Levels: L M S
```

通过因子水平的原始名称去改变因子水平的名称比通过位置改变更安全，因为你犯错误的机会更少（此处错误可能很难被发现）。而且，如果原始数据改变，因子水平的数值位置也可能会改变，这可能会对你的分析产生严重但是不易察觉的影响。

另见

如果需要改变字符向量的名称，参见 15.12 节。

15.11 去掉因子中不再使用的水平

问题

如何从一个因子中去除不再使用的水平？

方法

有的时候，在你对一堆数据进行操作之后，会有一些因子包含了不再使用的因子水平。下面就有一个例子：

```
sizes <- factor(c("small", "large", "large", "small", "medium"))
sizes <- sizes[1:3]
sizes

small  large  large
Levels: large medium small
```

为了删除这些不需要的水平，可以使用 droplevels() 函数：

```
sizes <- droplevels(sizes)
sizes

small  large  large
Levels: large small
```

讨论

droplevels() 函数保留了因子水平的顺序。另外，你可以使用 except 参数保留某个特定的水平。

15.12 在字符向量中改变元素的名称

问题

如何在字符向量中改变元素的名称？

方法

用 plyr 包中的 revalue() 函数或者 mapvalues() 函数：

```
sizes <- c("small", "large", "large", "small", "medium")
sizes

 "small" "large" "large" "small" "medium"

# 通过函数 revalue()，传递一组映射关系
sizes1 <- revalue(sizes, c(small="S", medium="M", large="L"))
sizes1

 "S" "L" "L" "S" "M"

# 也可以使用引号—— 如果原因子水平名称中存在空格等特殊字符，这将很有用
revalue(sizes, c("small"="S", "medium"="M", "large"="L"))

# mapvalues() 函数使用两组向量，而不是一组映射关系向量
mapvalues(sizes, c("small", "medium", "large"), c("S", "M", "L"))
```

讨论

在 R 中，更传统的方法是通过方括号索引去选择元素然后对它们重命名：

```
sizes <- c("small", "large", "large", "small", "medium")
sizes

 "small" "large" "large" "small" "medium"

sizes[sizes=="small"] <- "S"
sizes[sizes=="medium"] <- "M"
sizes[sizes=="large"] <- "L"

sizes

 "S" "L" "L" "S" "M"
```

另见

如果你要改变因子水平的名称，参见 15.10 节的方法。

15.13 把一个分类变量转化成另一个分类变量

问题

如何把一个分类变量转化成另一个分类变量？

方法

如下面的例子，我们使用 PlantGrowth 数据集的一个子数据集：

```
# 在 PlantGrowth 的一个子数据集上操作
pg <- PlantGrowth[c(1,2,11,21,22), ]
pg
```

```
weight group
  4.17  ctrl
  5.58  ctrl
  4.81  trt1
  6.31  trt2
  5.12  trt2
```

在这个例子中，我会把一个分类变量 group 编码到另一个分类变量 treatment 中。如果原先的值是 "ctrl"，新的值就是 "No"；如果原先的值是 "trt1" 或者 "trt2"，新的值就是 "Yes"。

这个可以用 match() 函数完成：

```
pg <- PlantGrowth

oldvals <- c("ctrl", "trt1", "trt2")
newvals <- factor(c("No", "Yes", "Yes"))

pg$treatment <- newvals[ match(pg$group, oldvals) ]
```

也可以使用向量索引的方法（但是比较笨拙）：

```
pg$treatment[pg$group == "ctrl"] <- "no"
pg$treatment[pg$group == "trt1"] <- "yes"
pg$treatment[pg$group == "trt2"] <- "yes"

# 转化为因子
pg$treatment <- factor(pg$treatment)
pg

weight group treatment
  4.17  ctrl        no
  5.58  ctrl        no
  4.81  trt1       yes
  6.31  trt2       yes
  5.12  trt2       yes
```

这里，我们把两个因子水平组合起来然后把结果存到一个新的列里面。如果你仅仅是想对因子水平重命名，参见 15.10 节的方法。

讨论

通过使用 & 和 | 操作符，编码准则同样可以基于多个列的取值：

```
pg$newcol[pg$group == "ctrl"  &  pg$weight <  5] <- "no_small"
pg$newcol[pg$group == "ctrl"  &  pg$weight >= 5] <- "no_large"
pg$newcol[pg$group == "trt1"] <- "yes"
pg$newcol[pg$group == "trt2"] <- "yes"

pg$newcol <- factor(pg$newcol)
pg
```

```
weight group weightcat treatment   newcol
  4.17  ctrl     small        no no_small
  5.58  ctrl     large        no no_large
  4.81  trt1     small       yes      yes
  4.17  trt1     small       yes      yes
  6.31  trt2     large       yes      yes
  5.12  trt2     large       yes      yes
```

我们也可以使用 interaction() 函数把数据框中的两列组合成一列。该函数会在两个值的中间加上一个 "." 符号。下面的例子把 weightcat 和 treatment 组合起来形成新的列 weighttrt：

```
pg$weighttrt <- interaction(pg$weightcat, pg$treatment)
pg
```

```
weight group weightcat treatment   newcol weighttrt
  4.17  ctrl     small        no no_small  small.no
  5.58  ctrl     large        no no_large  large.no
  4.81  trt1     small       yes      yes small.yes
  4.17  trt1     small       yes      yes small.yes
  6.31  trt2     large       yes      yes large.yes
  5.12  trt2     large       yes      yes large.yes
```

另见

对于更多关于改变因子水平名称的方法，参见 15.10 节。

对于把连续变量转变为分类变量，参见 15.14 节。

15.14 连续变量转变为分类变量

问题

如何把连续变量转变为分类变量？

方法

下面的例子，我们使用 PlantGrowth 数据集的一个子数据集。

```
# 在 PlantGrowth 的一个子数据集上操作
pg <- PlantGrowth[c(1,2,11,21,22), ]
pg
```

```
weight group
  4.17  ctrl
  5.58  ctrl
  4.81  trt1
  6.31  trt2
  5.12  trt2
```

在这个例子中，我们使用 cut() 函数把一个连续变量 weight 转化为分类变量 wtclass：

```
pg$wtclass <- cut(pg$weight, breaks = c(0, 5, 6, Inf))
pg

weight group wtclass
  4.17  ctrl   (0,5]
  5.58  ctrl   (5,6]
  4.81  trt1   (0,5]
  4.17  trt1   (0,5]
  6.31  trt2 (6,Inf]
  5.12  trt2   (5,6]
```

讨论

我们为三个类设定了四个边界值，边界值可以包括正无穷（Inf）和负无穷（-Inf）。如果一个值落在我们规定的区间之外，它的类别将被定为 NA（缺失值）。cut() 函数的输出结果是一个因子，你可以从下面这个例子中看出：因子水平的名称是以生成的区间命名的。

为了改变因子水平的名称，我们可以使用 cut() 中的 labels 参数：

```
pg$wtclass <- cut(pg$weight, breaks = c(0, 5, 6, Inf),
                  labels = c("small", "medium", "large"))

pg

 weight group wtclass
   4.17  ctrl   small
   5.58  ctrl  medium
   4.81  trt1   small
   4.17  trt1   small
   6.31  trt2   large
   5.12  trt2  medium
```

cut() 生成的区间是*左开右闭*的，换句话说，它们不会包含最小值，但是它们包含了最大值。对于值最小的一类，你可以通过设定参数 include.lowest=TRUE 使得它同时包含最小值和最大值。在这个例子中，这么做会使得 0 被包含到 small 类中；否则，0 会被赋为 NA。

如果你想让生成的区间是左闭右开的，设定参数 right=FALSE：

```
cut(pg$weight, breaks = c(0, 5, 6, Inf), right = FALSE)
```

另见

对于如何把一个分类变量转化成另一个分类变量，参见 15.13 节。

15.15 变量转换

问题

如何转换数据框中的变量？

方法

可以使用 $ 操作符来引用新列并对其赋予新值。在本例中，我们将使用 heightweight 数据集的复制版本：

```
library(gcookbook) # 为了使用数据集
# 复制数据集
hw <- heightweight
hw

sex ageYear ageMonth heightIn weightLb
  f   11.92      143     56.3     85.0
  f   12.92      155     62.3    105.0
...
  m   13.92      167     62.0    107.5
  m   12.58      151     59.3     87.0
```

这将会把 heightIn 的单位从英尺转换到厘米，然后存储到新的一列 heightCm 中：

```
hw$heightCm <- hw$heightIn * 2.54
hw

sex ageYear ageMonth heightIn weightLb heightCm
  f   11.92      143     56.3     85.0  143.002
  f   12.92      155     62.3    105.0  158.242
...
  m   13.92      167     62.0    107.5  157.480
  m   12.58      151     59.3     87.0  150.622
```

讨论

为了使代码更容易阅读，你可以使用 transform() 或 plyr 包中的 mutate() 函数。你仅需指定数据框一次，将其作为函数的第一个参数；它们提供了非常清晰的语法，尤其适合转换多个变量：

```
hw <- transform(hw, heightCm = heightIn * 2.54, weightKg = weightLb / 2.204)
library(plyr)
hw <- mutate(hw, heightCm = heightIn * 2.54, weightKg = weightLb / 2.204)
hw

sex ageYear ageMonth heightIn weightLb heightCm weightKg
  f   11.92      143     56.3     85.0  143.002 38.56624
  f   12.92      155     62.3    105.0  158.242 47.64065
...
  m   13.92      167     62.0    107.5  157.480 48.77495
  m   12.58      151     59.3     87.0  150.622 39.47368
```

也可以根据多个变量计算产生一个新的变量：

```
# 这些功能都一样：
hw <- transform(hw, bmi = weightKg / (heightCm / 100)^2)
```

```
hw <- mutate(hw, bmi = weightKg / (heightCm / 100)^2)
hw$bmi <- hw$weightKg / (hw$heightCm/100)^2
hw

sex ageYear ageMonth heightIn weightLb heightCm weightKg      bmi
  f   11.92      143     56.3     85.0  143.002 38.56624 18.85919
  f   12.92      155     62.3    105.0  158.242 47.64065 19.02542
...
  m   13.92      167     62.0    107.5  157.480 48.77495 19.66736
  m   12.58      151     59.3     87.0  150.622 39.47368 17.39926
```

transform() 和 mutate() 函数的最大区别是 transform() 会同时计算所有的新列，而 mutate() 将依次计算新列，这样在计算新列时就可以依赖之前的新列。由于 bmi 是由 heightCm 和 weightKg 计算来的，因此用 transform() 不能同时计算出这些变量；首先得计算 heightCm 和 weightKg，然后再计算 bmi，如上面代码所示。

使用 mutate() 函数时，我们可以一次完成这些计算。下面的代码和上面分开计算的代码效果一样：

```
hw <- heightweight
hw <- mutate(hw,
    heightCm = heightIn * 2.54,
    weightKg = weightLb / 2.204,
    bmi = weightKg / (heightCm / 100)^2)
```

另见

参阅 15.16 节来学习如何对分组数据进行转换。

15.16 按组转换数据

问题

如何根据分组变量来转换数据框中的数据？

方法

使用 plyr 包中的 ddply() 函数，在参数中调用 transform()，并指定运算：

```
library(MASS) # 为了使用数据集
library(plyr)
cb <- ddply(cabbages, "Cult", transform, DevWt = HeadWt - mean(HeadWt))

 Cult Date HeadWt VitC       DevWt
  c39  d16    2.5   51 -0.40666667
  c39  d16    2.2   55 -0.70666667
...
  c52  d21    1.5   66 -0.78000000
  c52  d21    1.6   72 -0.68000000
```

讨论

我们首先仔细观察 cabbages 数据集。它有两个分组变量（即因子），一个是 Cult，有两个水平 c39 和 c52，另一个是 Date，有三个水平：d16、d20 和 d21。它还有两个数值变量，HeadWt 和 VitC：

```
cabbages

 Cult Date HeadWt VitC
  c39  d16    2.5   51
  c39  d16    2.2   55
...
  c52  d21    1.5   66
  c52  d21    1.6   72
```

假设我们想知道每种情况下 HeadWt 和其整体均值的偏差。我们需要做的就是计算整体均值，然后在各个情况下减去它：

```
transform(cabbages, DevWt = HeadWt - mean(HeadWt))

 Cult Date HeadWt VitC        DevWt
  c39  d16    2.5   51 -0.093333333
  c39  d16    2.2   55 -0.393333333
 ...
  c52  d21    1.5   66 -1.093333333
  c52  d21    1.6   72 -0.993333333
```

很多时候，我们想对每个分组进行单独处理，这里的组由一个或多个分组变量指定。比如说，我们可能想这样标准化数据：求各个组的组内均值，然后求组内偏差，其中组是由 Cult 变量指定的。对于这种例子，可以使用 plyr 包中的 ddply() 函数，在参数中调用 transform()：

```
library(plyr)
cb <- ddply(cabbages, "Cult", transform, DevWt = HeadWt - mean(HeadWt))
cb

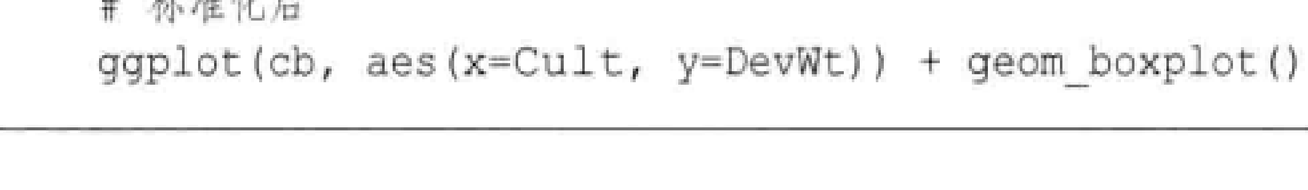
 Cult Date HeadWt VitC       DevWt
  c39  d16    2.5   51 -0.40666667
  c39  d16    2.2   55 -0.70666667
...
  c52  d21    1.5   66 -0.78000000
  c52  d21    1.6   72 -0.68000000
```

上面的代码首先会将 cabbages 数据根据 Cult 分割成几个独立的数据框。Cult 有两个水平，c39 和 c52，因此也就是分割成两个。然后在这两个数据框上使用 transform() 函数，其他参数保持不变。

注意，ddply() 函数和之前的 transform() 函数具有相同的参数列表。唯一的区别是 ddply() 略微调整了参数的位置，并且添加了分割变量（本例中是 Cult）。

标准化前、后的结果参见图 15-2。

```
# 标准化前
ggplot(cb, aes(x=Cult, y=HeadWt)) + geom_boxplot()
```



```
# 标准化后
ggplot(cb, aes(x=Cult, y=DevWt)) + geom_boxplot()
```

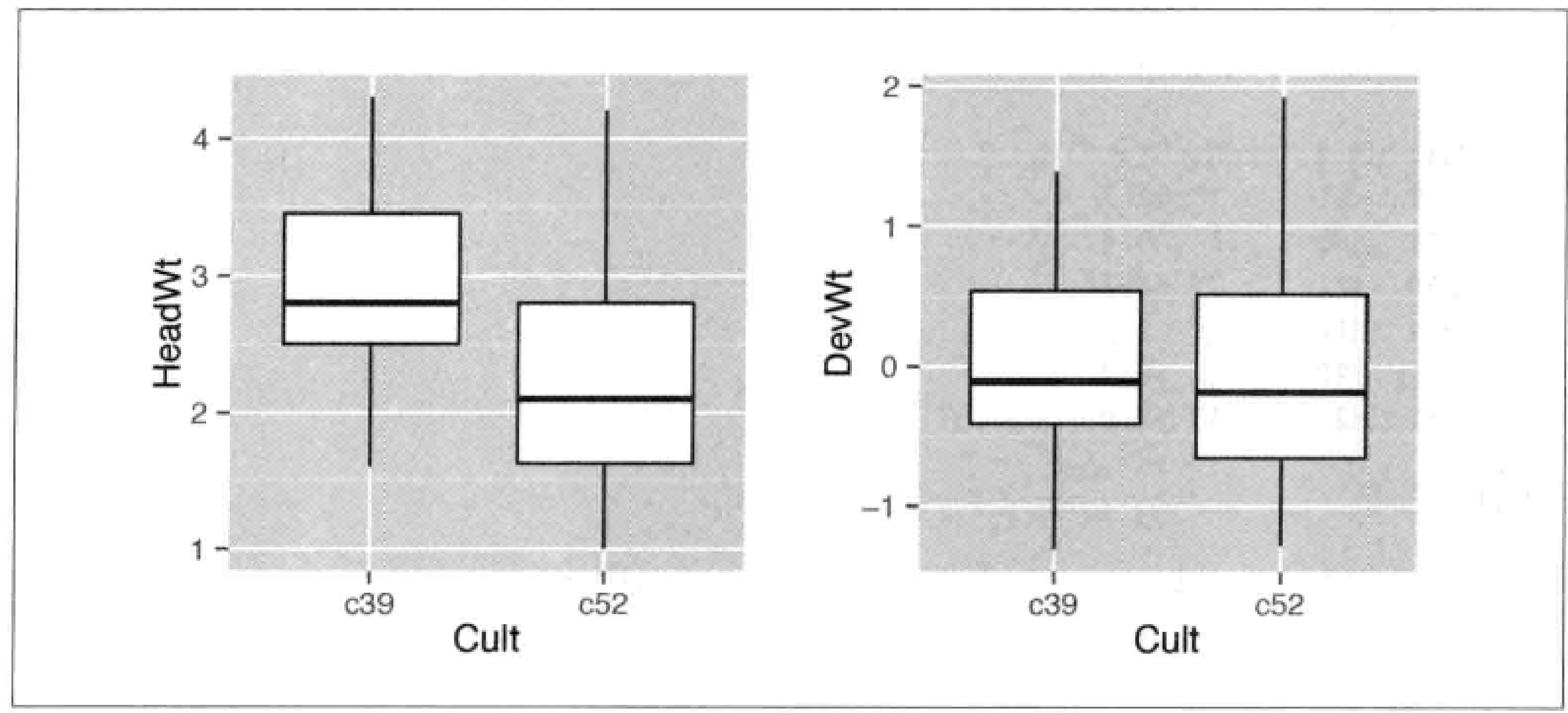


图 15-2　左图：标准化前　右图：标准化后

你也可以根据多个变量来分组、切割数据框，同时也可以在多个变量上进行运算。本例中将会根据 `Cult` 和 `Date` 切割数据，形成两者组合得到的分组，然后计算 `HeadWt` 和 `VitC` 在各个组的偏差：

```
ddply(cabbages, c("Cult", "Date"), transform,
      DevWt = HeadWt - mean(HeadWt), DevVitC = VitC - mean(VitC))
```

```
 Cult Date HeadWt VitC DevWt DevVitC
  c39  d16    2.5   51 -0.68     0.7
  c39  d16    2.2   55 -0.98     4.7
...
  c52  d21    1.5   66  0.03    -5.8
  c52  d21    1.6   72  0.13     0.2
```

另见

汇总分组数据可以参阅 15.17 节。

15.17　分组汇总数据

问题

如何对基于单个或多个变量分组的数据进行汇总？

方法

配合 summarise() 函数使用 plyr 包中的 ddply() 函数，并指定要进行的操作即可：

```
library(MASS) # 为了使用数据集
library(plyr)

ddply(cabbages, c("Cult", "Date"), summarise, Weight = mean(HeadWt),
      VitC = mean(VitC))

 Cult Date Weight VitC
  c39  d16   3.18 50.3
  c39  d20   2.80 49.4
  c39  d21   2.74 54.8
  c52  d16   2.26 62.5
  c52  d20   3.11 58.9
  c52  d21   1.47 71.8
```

讨论

我们先来仔细观察 cabbages 数据集。它有两个可以用来分组的因子：Cult，因子水平为 c39 和 c52；Date，因子水平为 d16、d20 和 d21。它还有两个数值变量：HeadWt 和 VitC：

```
Cabbages

 Cult Date HeadWt VitC
  c39  d16    2.5   51
  c39  d16    2.2   55
...
  c52  d21    1.5   66
  c52  d21    1.6   72
```

计算 HeadWt 的整体平均值是很简单的。我们仅仅需要对相应的列套用 mean() 函数即可，但接下来我们会发现，有时使用 summarise() 会更合适：

```
library(plyr)
summarise(cabbages, Weight = mean(HeadWt))

   Weight
 2.593333
```

得到的结果是一行一列的数据框，列名为 Weight。

我们经常会根据一个分组变量探索每个数据子集中的信息。比如，假设我们想找到每组 Cult 下的均值。为此，我们需要在 ddply() 中调用 summarise()。在我们同时使用它们时，注意参数是如何变化的：

```
library(plyr)
ddply(cabbages, "Cult", summarise, Weight = mean(HeadWt))
```

```
 Cult   Weight
 c39 2.906667
 c52 2.280000
```

上面的代码首先根据 Cult 的值将数据框 cabbages 切割成了几个小数据框。因子 Cult 有两个水平，c39 和 c52，因此也就有两个数据框。然后在每个数据框上套用 summarise() 函数，利用 mean() 函数计算每个数据框中 HeadWt 的均值并赋给新的列 Weight。结果就得到了两个一行的数据框，然后 ddply() 将它们合并为一个，正如结果所示。

根据多个变量（即多个列）切割数据框然后汇总也是很简单的：用一个包含多个变量名的向量即可。同样，可以对多个可计算的列进行汇总。这里我们根据 Cult 和 Date 分组，得到 HeadWt 和 VitC 在各组的均值：

```
ddply(cabbages, c("Cult", "Date"), summarise, Weight = mean(HeadWt),
      VitC = mean(VitC))

 Cult Date Weight VitC
  c39  d16   3.18 50.3
  c39  d20   2.80 49.4
  c39  d21   2.74 54.8
  c52  d16   2.26 62.5
  c52  d20   3.11 58.9
  c52  d21   1.47 71.8
```

除了求均值，我们还可以做很多其他的事。比方说，你可能想计算各个组的标准差和频数：使用 sd() 来计算标准差，length() 来计算频数：

```
ddply(cabbages, c("Cult", "Date"), summarise,
      Weight = mean(HeadWt),
      sd = sd(HeadWt),
      n = length(HeadWt))

 Cult Date Weight        sd  n
   c39 d16   3.18 0.9566144 10
   c39 d20   2.80 0.2788867 10
   c39 d21   2.74 0.9834181 10
   c52 d16   2.26 0.4452215 10
   c52 d20   3.11 0.7908505 10
   c52 d21   1.47 0.2110819 10
```

还有一些有用的函数也能够得到汇总统计量，包括 min()、max()、median() 等。

处理缺失值

一个可能的问题是：如果原数据中含有缺失值（NA），会导致输出的结果中也有。我们看看如果在 HeadWt 引入几个缺失值会发生什么？

```
c1 <- cabbages                 # 复制数据
c1$HeadWt[c(1,20,45)] <- NA   # 数据某些值赋值为 NA
```

```
ddply(c1, c("Cult", "Date"), summarise,
      Weight = mean(HeadWt),
      sd = sd(HeadWt),
      n = length(HeadWt))

Cult Date Weight        sd  n
  c39  d16     NA        NA 10
  c39  d20     NA        NA 10
  c39  d21   2.74 0.9834181 10
  c52  d16   2.26 0.4452215 10
  c52  d20     NA        NA 10
  c52  d21   1.47 0.2110819 10
```

这里有两个问题。第一个是如果任一输入值包含 NA，mean() 和 sd() 函数都会返回 NA。幸运的是，这些函数都有一个参数来处理这个问题：设置 na.rm=TRUE 即可忽略缺失值。

第二个问题是 length() 函数并没有对缺失值进行特殊处理，而是将它们视为“正常值”，但是这些都意味着缺失数据，因此它们不应该被计算在频数中。length() 函数并没有 na.rm 的选项，但是我们可以用 sum(!is.na(...)) 达到相同的效果。is.na() 返回一个逻辑向量：NA 返回 TRUE，非 NA 返回 FALSE。用 ! 取反后，再用 sum() 函数将 TRUE 的数量加起来。最终的结果就是非缺失值的频数：

```
ddply(c1, c("Cult", "Date"), summarise,
      Weight = mean(HeadWt, na.rm=TRUE),
      sd = sd(HeadWt, na.rm=TRUE),
      n = sum(!is.na(HeadWt)))

Cult Date   Weight        sd  n
  c39  d16 3.255556 0.9824855  9
  c39  d20 2.722222 0.1394433  9
  c39  d21 2.740000 0.9834181 10
  c52  d16 2.260000 0.4452215 10
  c52  d20 3.044444 0.8094923  9
  c52  d21 1.470000 0.2110819 10
```

组合缺失

如果在分组变量中有任何“空组合”，它们就不会出现在汇总的数据框中。缺失组合会给绘图带来麻烦。为了阐述这个问题，我们移除 c52 和 d21 的因子组合的所有样本点。图 15-3 左图就以柱状图的形式展示了组合缺失时的后果：

```
# 复制 cabbages 并移除同时包含 c52 和 d21 的行
c2 <- subset(c1, !( Cult=="c52" & Date=="d21" ) )

c2a <- ddply(c2, c("Cult", "Date"), summarise,
      Weight = mean(HeadWt, na.rm=TRUE),
      sd = sd(HeadWt, na.rm=TRUE),
      n = sum(!is.na(HeadWt)))
c2a
```

```
 Cult Date   Weight        sd  n
  c39  d16 3.255556 0.9824855  9
  c39  d20 2.722222 0.1394433  9
  c39  d21 2.740000 0.9834181 10
  c52  d16 2.260000 0.4452215 10
  c52  d20 3.044444 0.8094923  9
```

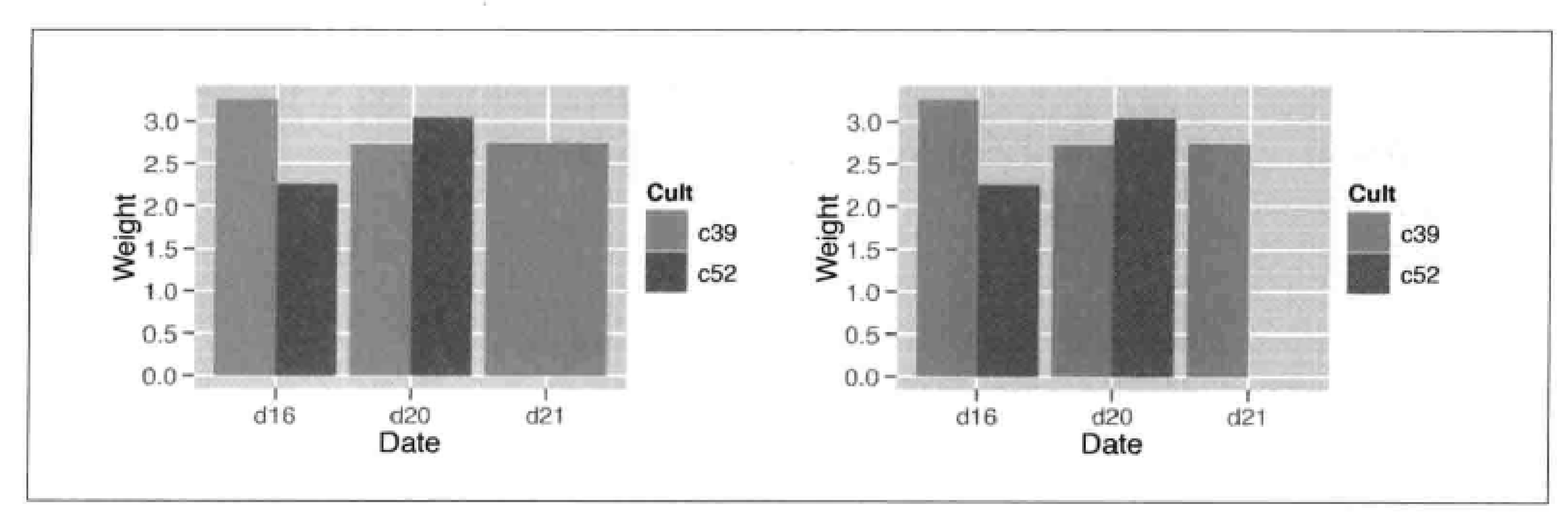


图 15-3 左图：有组合缺失的柱状图 右图：填充了缺失的组合

```
# 绘图
ggplot(c2a, aes(x=Date, fill=Cult, y=Weight)) + geom_bar(position="dodge")
```

为了填充缺失的组合（见图 15-3 右图），在 ddply() 函数中使用 .drop=FALSE 即可：

```
c2b <- ddply(c2, c("Cult", "Date"), .drop=FALSE, summarise,
      Weight = mean(HeadWt, na.rm=TRUE),
      sd = sd(HeadWt, na.rm=TRUE),
      n = sum(!is.na(HeadWt)))
c2b

 Cult Date   Weight        sd  n
  c39  d16 3.255556 0.9824855  9
  c39  d20 2.722222 0.1394433  9
  c39  d21 2.740000 0.9834181 10
  c52  d16 2.260000 0.4452215 10
  c52  d20 3.044444 0.8094923  9
  c52  d21      NaN        NA  0

# 绘图
ggplot(c2b, aes(x=Date, fill=Cult, y=Weight)) + geom_bar(position="dodge")
```

另见

如果你想计算标准误差和置信区间，参见 15.18 节。

6.8 节给出了一个使用 stat_summary() 计算均值并将均值堆叠放在图上的例子。

要对数据进行分组转换，参见 15.16 节。

15.18 使用标准误差和置信区间来汇总数据

问题

如何使用标准误差或置信区间来汇总数据？

方法

计算均值的标准误差包括两步：首先计算各组的标准差和频数，然后用这些值来计算得到标准误差。各组的标准误差就是标准差除以样本量的平方根。

```
library(MASS) # 为了使用数据集
library(plyr)

ca <- ddply(cabbages, c("Cult", "Date"), summarise,
      Weight = mean(HeadWt, na.rm=TRUE),
      sd = sd(HeadWt, na.rm=TRUE),
      n  = sum(!is.na(HeadWt)),
      se = sd/sqrt(n))
ca

 Cult Date Weight        sd  n         se
  c39  d16   3.18 0.9566144 10 0.30250803
  c39  d20   2.80 0.2788867 10 0.08819171
  c39  d21   2.74 0.9834181 10 0.31098410
  c52  d16   2.26 0.4452215 10 0.14079141
  c52  d20   3.11 0.7908505 10 0.25008887
  c52  d21   1.47 0.2110819 10 0.06674995
```



在 plyr 1.8 之前的版本中，summarise() 会同时创建所有的新列，因此你得在创建 sd 和 n 列之后再单独创建 se 列。

讨论

另外一种方法是在 ddply 函数内部计算标准误差。因为在 ddply 函数内部中不能引用 sd 和 n，因此我们得重新计算得到 se。下面的代码和之前展示的两步法效果是一样的：

```
ddply(cabbages, c("Cult", "Date"), summarise,
      Weight = mean(HeadWt, na.rm=TRUE),
      sd = sd(HeadWt, na.rm=TRUE),
      n  = sum(!is.na(HeadWt)),
      se = sd / sqrt(n))
```

置信区间

置信区间是通过均值的标准误差和自由度计算得到的。要计算置信区间，首先使用 qt() 函数得到分位数，然后和标准误差相乘即可。给定概率值和自由度，qt() 函数会计算出对应 t 分布的分位数。对 95% 的置信区间来说，应该使用 0.975 的概率值；

对钟形的 t 分布，这对应了曲线两端各 2.5% 的面积。自由度是样本量大小减去 1。

下面的代码将会计算每组标准误差的乘数。由于一共有 6 个组并且每组都有 10 个观测值，因此它们有相同的乘数。

```
ciMult <- qt(.975, ca$n-1)
ciMult

# 2.262157 2.262157 2.262157 2.262157 2.262157 2.262157
```

现在我们可以将上面的向量乘以标准误差来得到 95% 的置信区间：

```
ca$ci <- ca$se * ciMult

 Cult  Date Weight        sd  n          se        ci
  c39   d16   3.18 0.9566144 10 0.30250803 0.6843207
  c39   d20   2.80 0.2788867 10 0.08819171 0.1995035
  c39   d21   2.74 0.9834181 10 0.31098410 0.7034949
  c52   d16   2.26 0.4452215 10 0.14079141 0.3184923
  c52   d20   3.11 0.7908505 10 0.25008887 0.5657403
  c52   d21   1.47 0.2110819 10 0.06674995 0.1509989
```

我们可以一步完成上面的事情，如下：

```
ca$ci95 <- ca$se * qt(.975, ca$n)
```

对 99% 的置信区间，使用 0.995 的概率值。

误差条表示均值的标准误差，它和置信区间有相同的功能：给看图的人展示估计总体均值的好坏程度。标准误差是抽样分布的标准差。置信区间更容易解释。很粗略地来说，95% 的置信区间意味着总体均值有 95% 的机率落在区间中（实际上，它的含义并不是这样，但是这貌似简单的话题很难在这里解释；如果你想深入了解，请阅读贝叶斯统计学）。

这个函数会一并计算标准差、频数、标准误差和置信区间。并且它还可以处理缺失值和空缺组合，此时只需设置 na.rm 和 .drop 选项即可。函数默认计算的是 95% 的置信区间，这个也可以通过 conf.interval 参数来改变。

```
summarySE <- function(data=NULL, measurevar, groupvars=NULL,
                      conf.interval=.95, na.rm=FALSE, .drop=TRUE) {
    require(plyr)

    # 新版本的 length 可以处理缺失值： 如果 na.rm==T，则排除缺失值
    length2 <- function (x, na.rm=FALSE) {
        if (na.rm) sum(!is.na(x))
        else       length(x)
    }

    # 汇总
    datac <- ddply(data, groupvars, .drop=.drop,
                   .fun = function(xx, col, na.rm) {
                           c( n    = length2(xx[,col], na.rm=na.rm),
                              mean = mean   (xx[,col], na.rm=na.rm),
```

```
                                sd    = sd     (xx[,col], na.rm=na.rm)
                                )
                            },
                    measurevar,
                    na.rm
            )


    # 重命名 "mean" 列
    datac <- rename(datac, c("mean" = measurevar))

    datac$se <- datac$sd / sqrt(datac$n)  # 计算均值的标准误差

    # 标准误差的置信区间乘数
    # 为置信区间计算 t 统计量：
    # 比如，如果 conf.interval 是 .95，就使用 .975 （上 / 下），并且
    # 使用 df=n-1，或如果 n==0，则 df=0
    ciMult <- qt(conf.interval/2 + .5, datac$n-1)
    datac$ci <- datac$se * ciMult

    return(datac)
}
```

这里的应用例子计算 99% 的置信区间，并且可以处理缺失值和空缺组合：

```
# 移除 c52 和 d21 对应的所有行
c2 <- subset(cabbages, !( Cult=="c52" & Date=="d21" ) )

# 将一些值设置为 NA
c2$HeadWt[c(1,20,45)] <- NA

summarySE(c2, "HeadWt", c("Cult", "Date"), conf.interval=.99,
          na.rm=TRUE, .drop=FALSE)

 Cult Date  n   HeadWt        sd         se        ci
  c39  d16  9 3.255556 0.9824855 0.32749517 1.0988731
  c39  d20  9 2.722222 0.1394433 0.04648111 0.1559621
  c39  d21 10 2.740000 0.9834181 0.31098410 1.0106472
  c52  d16 10 2.260000 0.4452215 0.14079141 0.4575489
  c52  d20  9 3.044444 0.8094923 0.26983077 0.9053867
  c52  d21  0      NaN        NA         NA        NA
Warning message:
In qt(p, df, lower.tail, log.p) : NaNs produced
```

有空缺组合的时候，程序会给出警告信息。这并不是一个问题，只是告诉我们它不能计算一个没有观测值的组的分位数。

另见

使用这里计算的值并添加误差条到一个图上，参见 7.7 节。

15.19 把数据框从“宽”变“长”

问题

如何把把数据框从“宽”变“长”?

方法

使用 reshape2 包中的 melt() 函数。在 anthoming 数据集中，angle 表示蚂蚁行走方向与家的方向的角度（正表示顺时针），每个 angle 有两个度量变量（measurement）描述它，expt 表示在实验条件下走这个方向的蚂蚁数量，ctrl 表示在控制条件下走这个方向的蚂蚁数量：

```
library(gcookbook) # 为了使用数据
anthoming

 angle expt ctrl
   -20    1    0
   -10    7    3
     0    2    3
    10    0    3
    20    0    1
```

我们可以重塑该数据的结构，把两个度量变量都放在单独一列中。具体的做法是把这两列的值放在一列中并新增一列存放两列的名称：

```
library(reshape2)
melt(anthoming, id.vars="angle", variable.name="condition", value.name="count")

 angle condition count
   -20      expt     1
   -10      expt     7
     0      expt     2
    10      expt     0
    20      expt     0
   -20      ctrl     0
   -10      ctrl     3
     0      ctrl     3
    10      ctrl     3
    20      ctrl     1
```

两个数据框包含了一样的信息，但是第二种组织方式在某些情况下更有利于我们分析。

讨论

在原数据中，有标识变量（ID variable）和度量变量（measure variable）。标识标量表明哪些值要汇集到一起，即哪些值是在描述同一个对象。在原数据中，第一行有两个度量变量来描述 angle 是 -20 的情况。在变换后的数据中，两个度量变量 expt 和 ctrl 不再出现在同一行中，但是我们仍然可以看出它们都在描述同一个 angle，因为它们的 angle 值是一样的。

度量变量会被默认为除标识变量以外的所有变量。这些度量变量的名称会被放到一个叫 variable.name 的列中，而它们对应的取值则是放到一个名为 value.name 的列中。

如果你不想用所有的非标识变量作为度量变量，可以指明哪些变量是你需要的（参数 measure.vars）。例如，在 drunk 数据集中，我们可以只用 0-29 和 30-39 这两组：

```
drunk
    sex 0-29 30-39 40-49 50-59 60+
   male  185   207   260   180  71
 female    4    13    10     7  10

melt(drunk, id.vars="sex", measure.vars=c("0-29", "30-39"),
     variable.name="age", value.name="count")

    sex   age count
   male  0-29   185
 female  0-29     4
   male 30-39   207
 female 30-39    13
```

同样，也可以用多列作为标识变量：

```
plum_wide

 length      time dead alive
   long   at_once   84   156
   long in_spring  156    84
  short   at_once  133   107
  short in_spring  209    31

melt(plum_wide, id.vars=c("length","time"), variable.name="survival",
                           value.name="count")

 length      time survival count
   long   at_once     dead    84
   long in_spring     dead   156
  short   at_once     dead   133
  short in_spring     dead   209
   long   at_once    alive   156
   long in_spring    alive    84
  short   at_once    alive   107
  short in_spring    alive    31
```

有些数据集并不具备标识变量。例如，在 corneas 数据集中，每一行代表了一对度量变量，但是这里没有标识变量。没有标识变量，你无法知道哪些值是在描述同一个对象。这种情况下，你可以在用 melt() 函数之前给这个数据框添加一个标识变量。

```
# 创造数据的一个副本
co <- corneas
co

affected notaffected
     488         484
     478         478
```

```
     480          492
     426          444
     440          436
     410          398
     458          464
     460          476
```

```
# 添加标识列
co$id <- 1:nrow(co)

melt(co, id.vars="id", variable.name="eye", value.name="thickness")
```

```
 id         eye thickness
  1    affected       488
  2    affected       478
  3    affected       480
  4    affected       426
  5    affected       440
  6    affected       410
  7    affected       458
  8    affected       460
  1 notaffected       484
  2 notaffected       478
  3 notaffected       492
  4 notaffected       444
  5 notaffected       436
  6 notaffected       398
  7 notaffected       464
  8 notaffected       476
```

用数值作为标识变量可能会给后续分析带来问题，所以你可能要用 as.character() 函数把它转化成字符型的向量或者用 factor() 将其转化为因子。

另见

从“长”到“宽”的转化，参见 15.20 节。

stack() 函数也可以把数据框从“宽”变“长”，请参见 R 的帮助文档。

15.20 把数据框从“长”变“宽”

问题

如何把数据框从“长”变“宽”?

方法

使用 reshape2 包中的 dcast() 函数。在这个例子中，我们使用 plum 数据集，它是以长格式进行存储的。

```
library(gcookbook) # 为了使用数据集
plum
```

```
 length      time survival count
   long   at_once     dead    84
   long in_spring     dead   156
  short   at_once     dead   133
  short in_spring     dead   209
   long   at_once    alive   156
   long in_spring    alive    84
  short   at_once    alive   107
  short in_spring    alive    31
```

从“长”到“宽”的转化把一列中不重复的值提取出来并用它们作为新列的名称，然后用另一列作为新列的数据源。例如，我们可以把 survival 中的值放到表格顶端，然后用 count 中的值去进行填充：

```
library(reshape2)
dcast(plum, length + time ~ survival, value.var="count")
```

```
 length      time dead alive
   long   at_once   84   156
   long in_spring  156    84
  short   at_once  133   107
  short in_spring  209    31
```

讨论

dcast() 函数要求你指明标识变量（留下来的列）和可变变量（variable variable）（会转化成新生成列的变量）。这个步骤用公式完成，波浪线（~）左边表示标识变量，右边表示可变变量。

前面的例子有两个标识变量和一个可变变量。下一个例子中只有一个标识变量但是有两个可变变量。当有多个可变变量时，生成的列名由下划线连接起来：

```
dcast(plum, time ~ length + survival, value.var="count")
```

```
      time long_dead long_alive short_dead short_alive
   at_once        84        156        133         107
 in_spring       156         84        209          31
```

另见

数据从“宽”变“长”，参见 15.19 节。

unstack() 函数也可以把数据框从“长”变“宽”，参见 R 的帮助文档。

15.21 把时间序列数据对象拆分成时间和数据

问题

如何把时间序列对象拆分成观测时间和每个时点的观测数据？

方法

time() 函数可以得到每个观测的时间值，然后用 as.numeric() 函数将时间和该数据转化为数值形式：

```
# 查看时间序列对象 nhtemp
nhtemp

Time Series:
Start = 1912
End = 1971
Frequency = 1
 [1] 49.9 52.3 49.4 51.1 49.4 47.9 49.8 50.9 49.3 51.9 50.8 49.6 49.3 50.6 48.4
[16] 50.7 50.9 50.6 51.5 52.8 51.8 51.1 49.8 50.2 50.4 51.6 51.8 50.9 48.8 51.7
[31] 51.0 50.6 51.7 51.5 52.1 51.3 51.0 54.0 51.4 52.7 53.1 54.6 52.0 52.0 50.9
[46] 52.6 50.2 52.6 51.6 51.9 50.5 50.9 51.7 51.4 51.7 50.8 51.9 51.8 51.9 53.0

# 得到每次观测的时间
as.numeric(time(nhtemp))

 [1] 1912 1913 1914 1915 1916 1917 1918 1919 1920 1921 1922 1923 1924 1925 1926
[16] 1927 1928 1929 1930 1931 1932 1933 1934 1935 1936 1937 1938 1939 1940 1941
[31] 1942 1943 1944 1945 1946 1947 1948 1949 1950 1951 1952 1953 1954 1955 1956
[46] 1957 1958 1959 1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 1970 1971

# 得到每次观测的值
as.numeric(nhtemp)

 [1] 49.9 52.3 49.4 51.1 49.4 47.9 49.8 50.9 49.3 51.9 50.8 49.6 49.3 50.6 48.4
[16] 50.7 50.9 50.6 51.5 52.8 51.8 51.1 49.8 50.2 50.4 51.6 51.8 50.9 48.8 51.7
[31] 51.0 50.6 51.7 51.5 52.1 51.3 51.0 54.0 51.4 52.7 53.1 54.6 52.0 52.0 50.9
[46] 52.6 50.2 52.6 51.6 51.9 50.5 50.9 51.7 51.4 51.7 50.8 51.9 51.8 51.9 53.0

# 把它们放进一个数据框中
nht <- data.frame(year=as.numeric(time(nhtemp)), temp=as.numeric(nhtemp))
Nht

 year temp
 1912 49.9
 1913 52.3
 ...
 1970 51.9
 1971 53.0
```

讨论

当观测时间是有规律的时间区间时，时间序列对象能有效地存储信息，但是为了使用 ggplot2，它们需要被拆分成观测时间和观测数据。

有些时间序列对象是周期性的。例如，presidents 数据集，每一年有四个观测，一个季度一次观测：

```
presidents

     Qtr1 Qtr2 Qtr3 Qtr4
1945   NA   87   82   75
1946   63   50   43   32
1947   35   60   54   55
 ...
1972   49   61   NA   NA
1973   68   44   40   27
1974   28   25   24   24
```

为了把它转化为有两列值的数据框，其中一列用分数表示年，其步骤和前面的一样：

```
pres_rating <- data.frame(
    year     = as.numeric(time(presidents)),
    rating   = as.numeric(presidents)
)
pres_rating

    year rating
 1945.00     NA
 1945.25     87
 1945.50     82
 ...
 1974.25     25
 1974.50     24
 1974.75     24
```

我们也可以把年和季度分别存在两列，这可能对于某些可视化方法会有所帮助：

```
pres_rating2 <- data.frame(
    year     = as.numeric(floor(time(presidents))),
    quarter  = as.numeric(cycle(presidents)),
    rating   = as.numeric(presidents)
)
pres_rating2

 year quarter rating
 1945       1     NA
 1945       2     87
 1945       3     82
 ...
 1974       2     25
 1974       3     24
 1974       4     24
```

另见

zoo 包在处理时间序列对象的时候也非常有用。

附录A

ggplot2介绍

本书的很多内容都涉及 Hadley Wickham 编写的 ggplot2 包。ggplot2 的出现只有几年时间，但就在这短短的时间之内，它已经吸引了 R 社区内的众多用户，这无不归因于它的功能丰富、有着清晰一致的接口以及美观的输出。

相比 R 中的其他绘图包，ggplot2 采取了一种不同的制图方式。它得名于 Leland Wilkinson 的《Grammar of Graphics》（图形的语法）一书，这本书为描述数据类图形提供了一套形式化的、结构化的观点。

尽管本书很多内容是围绕 ggplot2 写成，但这并不意味着它就是 R 图形的终极归宿。举例来说，有时候我们会发现使用 R 基础图形来观察和探索数据更快更简单，特别是在数据尚未被合适地结构化以配合 ggplot2 使用之前。有些任务 ggplot2 却无法完成，或是无法像其他的绘图包做得那样好。有些其他的任务 ggplot2 虽然可以胜任，但却更适合用某些特定的包来处理。不过在大多数场合，我相信 ggplot2 可以对投入的时间给出最好的回报，而且它提供了美观、具备出版级质量的输出结果。

另一个优秀的通用图形包是 Deepayan Sarkar 的 lattice 包，这是一套格子（trellis）图形的实现，它包含在 R 的基础安装中。

如果你想对 ggplot2 有更深入的理解，请继续阅读吧！

A.1 背景知识

在数据图形中，存在着一种从数据属性到视觉属性的映射（或者说对应）。数据属性通常是数值或类别，而视觉属性则包括点的 x 和 y 坐标、线条颜色、条形的高度等。不将数据映射为视觉属性的可视化并不能算是数据可视化。从表面上看，使用 x 轴上的坐标来表示一个数值，可能看起来和使用某种带颜色的点来表示一个数值有很大的不同，但是在抽象的层面上，它们是一样的。只要是做过数据图形的人，至少会对这

一点有一种模糊的认识。对于大多数人来说，这种认识也就停留在这个阶段。

对于“图形的语法”理论来说，这种深刻的相似性不单单得到了承认，而且更是处于中心地位。在 R 的基础绘图函数中，每种数据属性到视觉属性的映射都仅仅是单个函数本身的特例，而且映射的变更可能需要重构数据，或是键入完全不同的绘图命令，或者两者皆有。

为了说明这一点，下面将会展示一幅根据 gcookbook 包中的 simpledat 数据集绘制的图形：

```
library(gcookbook)  # 为了使用数据集
simpledat

   A1 A2 A3
B1 10  7 12
B2  9 11  6
```

此数据可以绘制一幅简单的分组条形图，各 A 值沿 x 轴排布，条形依 B 值分组（见图 A-1）：

```
barplot(simpledat, beside=TRUE)
```

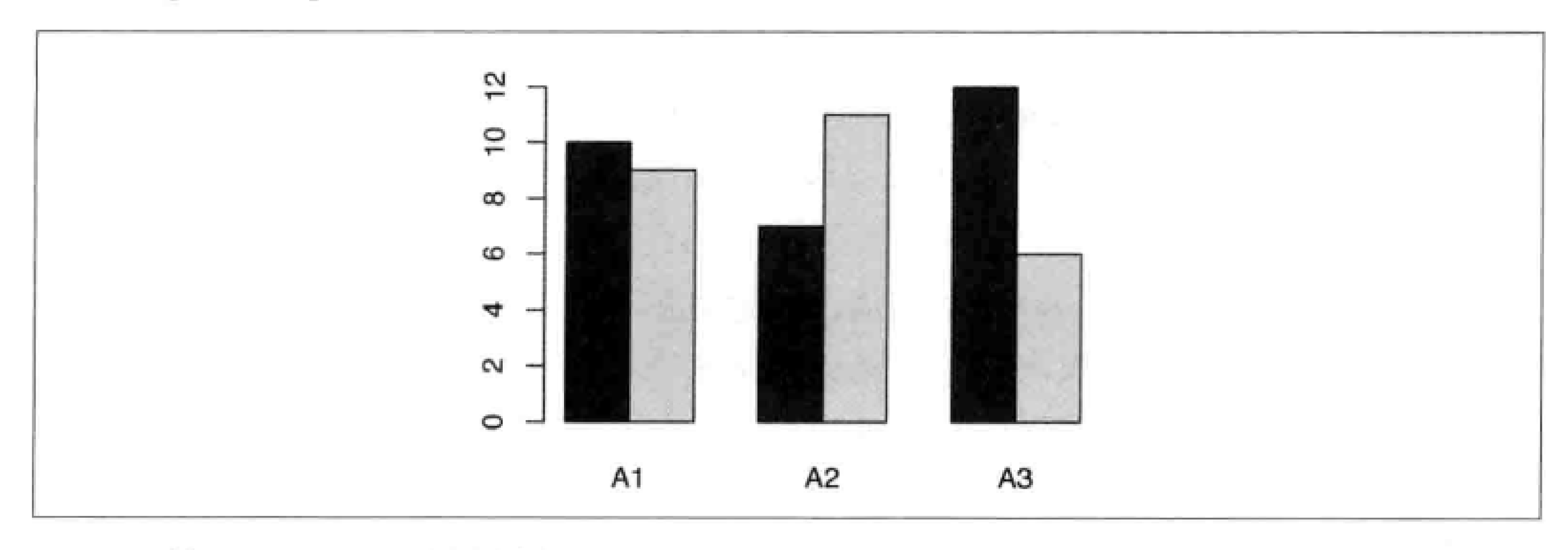


图 A-1　使用 barplot() 绘制的条形图

我们可能想做的一件事就是，调换数据值的位置，使得 B 值沿 x 轴排布而 A 值用来分组。要这样做，我们需要转置矩阵以重构数据：

```
t(simpledat)

   B1 B2
A1 10  9
A2  7 11
A3 12  6
```

使用重构后的数据，我们就可以像前面一样创建图形了（见图 A-2）：

```
barplot(t(simpledat), beside=TRUE)
```

另一方面，我们可能想使用线条而不是条形来呈现数据，如图 A-3 所示。要使用基础图形完成这项任务，我们需要使用一套完全不同的命令。首先，我们需要调用 plot() 来通

知 R 创建一幅新图并为其中一行数据绘制一条线，然后使用 lines() 去绘制另一行数据：

```
plot(simpledat[1,], type="l")
lines(simpledat[2,], type="l", col="blue")
```

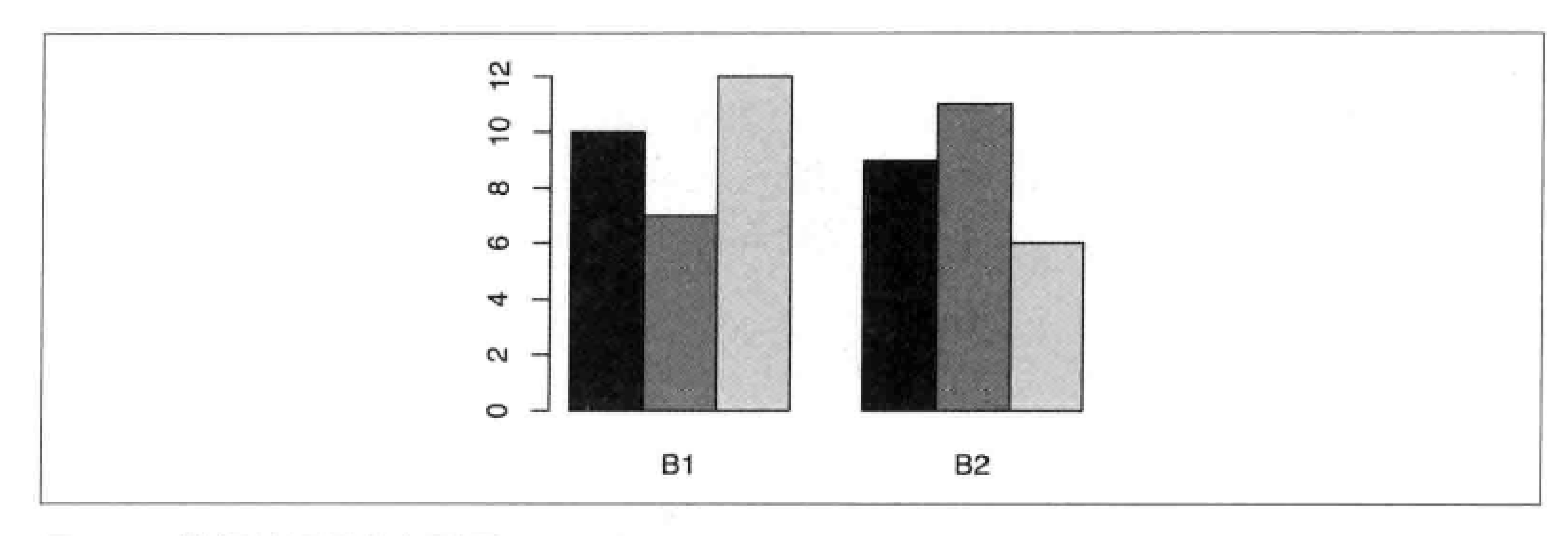


图 A-2　数据转置后的条形图

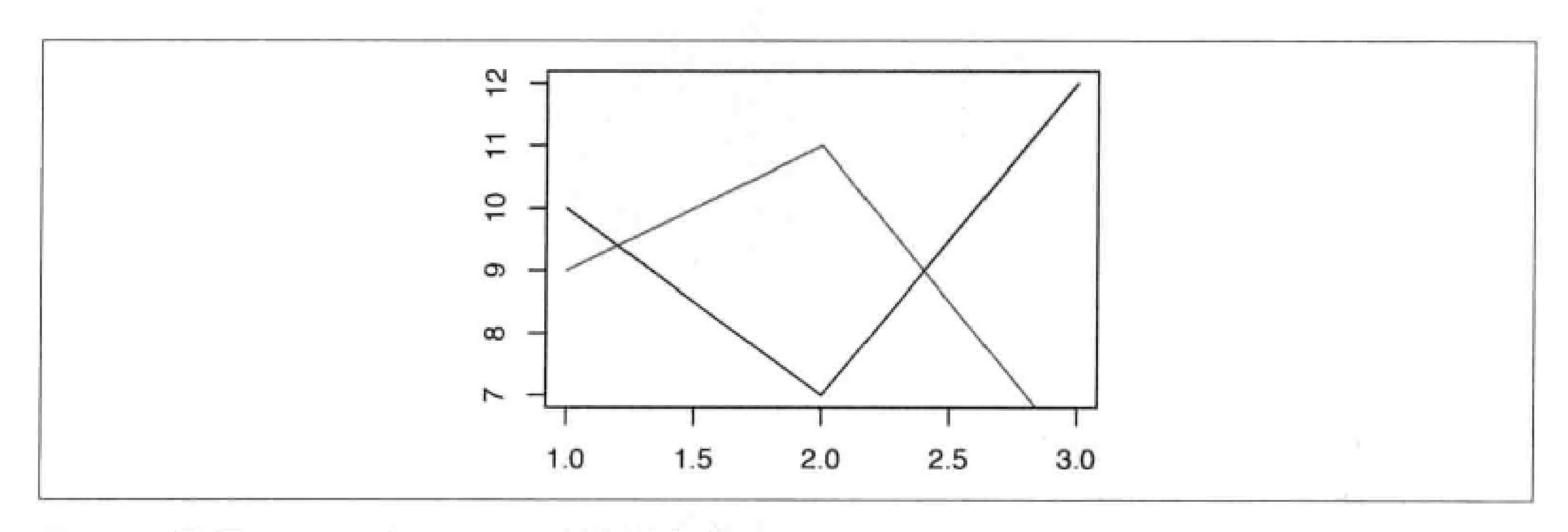


图 A-3　使用 plot() 和 lines() 绘制的折线图

结果图中有一些怪异的地方。第二条（蓝色）线的下部超出了可见范围，这是因为 *y* 轴的范围是在调用 plot() 函数时仅为第一条线设定的。另外，*x* 轴是数值型而不是类别型。

现在让我们来看看对应的 ggplot2 代码和图形。对于 ggplot2 来说，数据的结构是一成不变的：它要求的是“长”格式的数据框，而不是之前使用的相反的“宽”格式。当数据为长格式时，每行表示一个条目。其所属的分组不由它们在矩阵中的位置决定，而是在一个单独的列中指定。这是将 simpledat 转换为长格式的结果：

```
simpledat_long

 Aval Bval value
   A1   B1    10
   A1   B2     9
   A2   B1     7
   A2   B2    11
   A3   B1    12
   A3   B2     6
```

这种格式使用了不同的结构，表示的却是相同的信息。长格式有利有弊，但总的来说，我发现在处理复杂数据集的时候它能让事情变得简单一些。关于宽格式和长格式之间的转换方法，参见 15.19 节和 15.20 节。

要绘制第一幅分组条形图（见图 A-4），我们首先需要加载 ggplot2 包。然后使用 x=Aval 将 Aval 映射至 *x* 轴，使用 fill=Bval 将 Bval 映射为填充色。这样就使得各 A 值沿 *x* 轴分布，而让 B 值决定分组。我们还需要使用 y=value 将 value 映射到 *y* 的位置，即条形的高度。最后，我们使用 geom_bar() 来绘制条形（别担心其他细节，我们会很快讲到）：

```
library(ggplot2)
ggplot(simpledat_long, aes(x=Aval, y=value, fill=Bval)) +
    geom_bar(stat="identity", position="dodge")
```

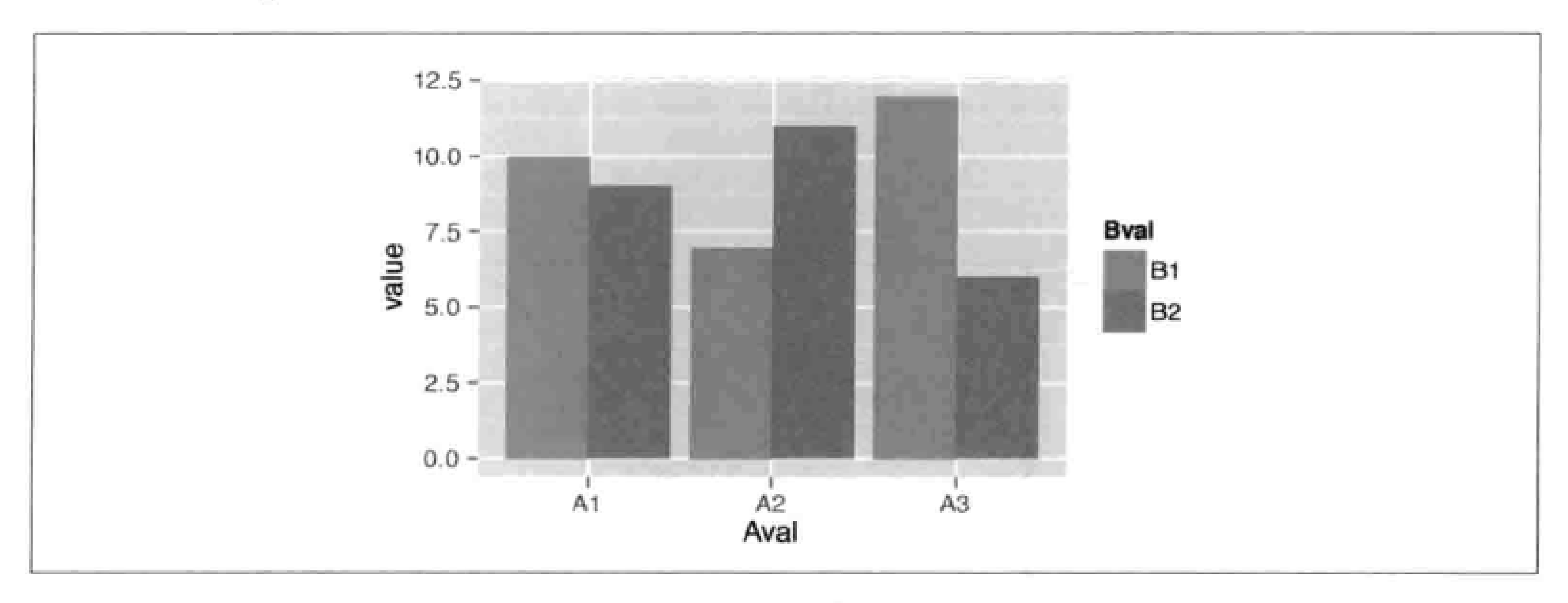


图 A-4　使用 **ggplot()** 和 **geom_bar()** 绘制的条形图

要调换位置，令 B 值沿 *x* 轴排布而让 A 值决定分组（见图 A-5），只需简单地交换映射参数，使得 x=Bval 且 fill=Aval 即可。与基础图形不同的是，这里无需修改数据，仅需修改绘图命令：

```
ggplot(simpledat_long, aes(x=Bval, y=value, fill=Aval)) +
    geom_bar(stat="identity", position="dodge")
```

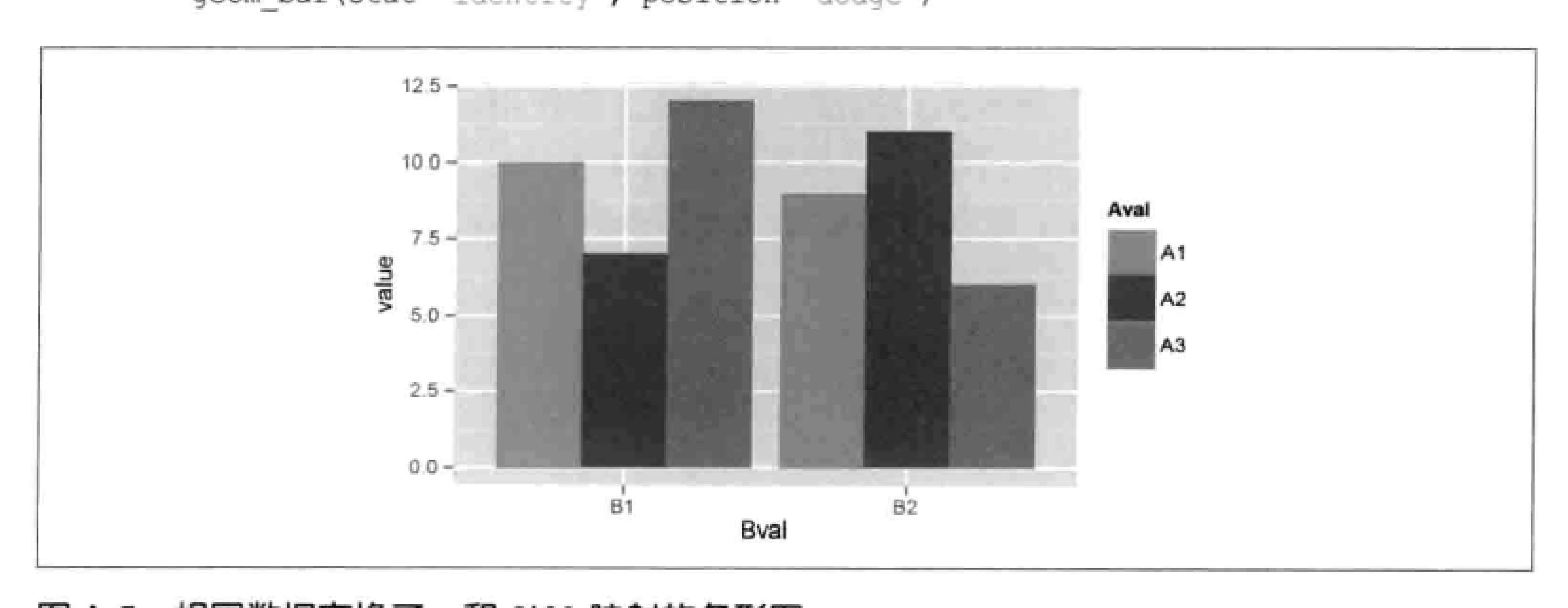


图 A-5　相同数据交换了 **x** 和 **fill** 映射的条形图



你可能已经注意到了，在 ggplot2 中，各个图形部件都是通过操作符 + 进行组合的。你可以通过添加组件的方式来渐进式地构建一个 ggplot 对象，在做完以后，就可以打印出来了。

要将图 A-5 所示的图修改为折线图（见图 A-6），需要把 `geom_bar()` 修改为 `geom_line()`。同时使用参数 `colour` 将 `Bval` 映射为线条的颜色而非填充的颜色（注意参数名称的英式拼法——因为ggplot2的作者是新西兰人）。同样，我们仍然无需担心其他细节：

```
ggplot(simpledat_long, aes(x=Aval, y=value, colour=Bval, group=Bval)) +
    geom_line()
```

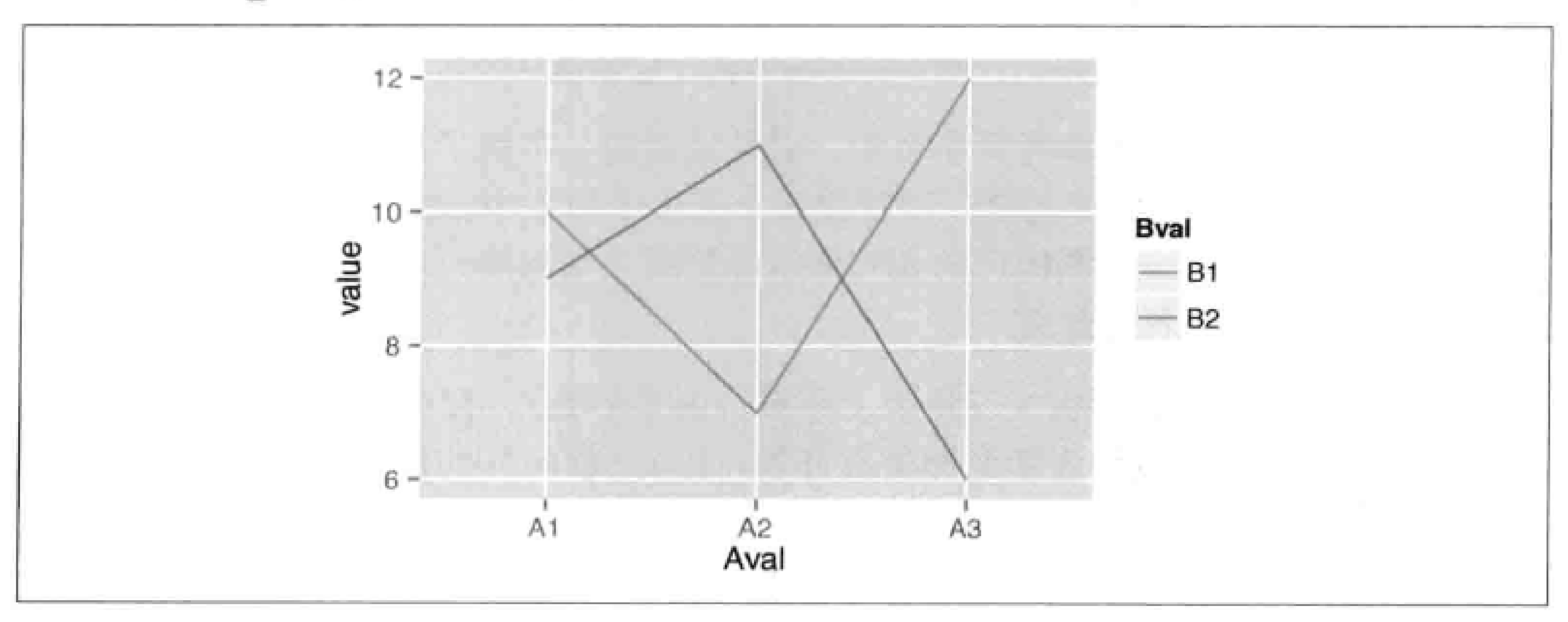


图 A-6　使用 `ggplot()` 和 `geom_line()` 绘制的折线图

在结合基础图形将条形图改为折线图时，我们不得不使用完全不同的命令。但通过 ggplot2，我们仅需将几何对象从条形改为线条。结果图也与基础图形版有着重要区别：由于所有的线条都是同时绘制而非每次绘制一条，所以 y 的范围已被自动调整以适应全体数据，且 x 轴仍保持为类别型而不会被转换为数值型。ggplot2 图形还带有自动生成的图例。

A.2　若干术语和理论

在更进一步之前，先定义一些 ggplot2 中使用的术语会有所帮助。

- 数据（data）是我们想要可视化的对象。其中包含了变量（variable），变量存储于数据框的每一列。
- 几何对象（geom）是用以呈现数据的几何图形对象，如条形、线条和点。
- 图形属性（aesthetic）是几何对象的视觉属性，如 x 坐标和 y 坐标、线条颜色、点的形状等。
- 数据的值和图形属性之间存在着某类映射（mapping）。
- 标度（scale）控制着数据空间的值到图形属性空间的值的映射。一个连续型的 y 标度会将较大的数值映射至空间中纵向更高的位置。
- 引导元素（guide）向看图者展示了如何将视觉属性映射回数据空间。最常用的引导

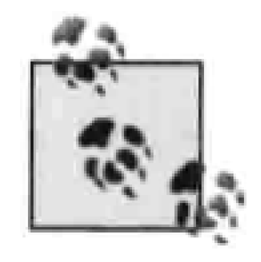
元素是坐标轴上的刻度线和标签。

要了解一个典型的映射是如何工作的，可以来看一个例子。我们拥有数据，也就是一组数值或类别值。有几何对象来表示每个观测。我们有图形属性，如 *y*(纵向）的位置。还有一种标度定义了从数据空间（数值）到图形属性空间（纵向位置）的映射。一个典型的线性 *y* 标度可将数值 0 映射到图形的基线，将 5 映射到中间，将 10 映射到顶端。一个对数型 *y* 标度则会以不同的方式放置它们。

以上并不是仅有的数据空间和图形属性空间。在抽象的图形语法层面，数据和图形属性可以是任何东西；在 ggplot2 的实现中，有一些预定义的数据和图形属性类型。常用的数据类型包括数值、类别值和字符串。一些常用的图形属性包括横纵位置、颜色、大小和形状。

要解读图形，看图者需要参考引导元素。引导元素的一个例子是包含刻度线和标签的 *y* 轴。看图者参考这个引导元素，以解读某个点位于标度中间所表示的含义。图例则是另一类引导元素。一个图例可向人们展示环形或三角形的点所代表的含义，或者蓝色或红色的线条所代表的含义。

某些图形属性，如点的形状（三角形、环形、正方形等），仅对类别型变量有效。某些图形属性则对类别型和连续型变量都有效，如 *x*（横向）的位置。对于条形图，变量必须为类别型——*x* 轴上放一个连续型变量是毫无意义的。对于散点图，变量必须为数值型。这些数据类型（类别型和数值型）均可被映射到 *x* 位置的图形属性空间，但它们需要不同类型的标度。



在 ggplot2 术语中，类别型变量被称为离散型变量，数值型变量则叫做连续型变量。这些术语可能并不总是与别处的用法相对应。在 ggplot2 意义下的连续型变量有时是常规意义下的离散型变量。例如，可见太阳黑子的数量必为一个整数，所以它属于数值型（对于 ggplot2 来说是连续型）和（在日常用语中的）离散型。

A.3 构建一幅简单图形

ggplot2 对数据结构的要求很简单：数据必须存储于数据框中，且每类被映射为某种图形属性的变量必须独立集中存储在一列。在前文的 `simpledat` 示例中，我们首先将一个变量映射为图形属性 `x`，将另一个变量映射为图形属性 `fill`；然后改变了变量到图形属性的映射规则。

我们将在这里演练一个简单的示例。首先，创建一个含有一些样本数据的数据框：

```
dat <- data.frame(xval=1:4, yval=c(3,5,6,9), group=c("A","B","A","B"))
dat

xval  yval  group
   1     3      A
```

```
2     5     B
3     6     A
4     9     B
```

一个基本的 ggplot() 调用范式看起来就像这样：

```
ggplot(dat, aes(x=xval, y=yval))
```

这样就使用数据框 dat 创建了一个 ggplot 对象。同时也在 aes() 中指定了默认的图形属性映射：

- x=xval 将列 xval 映射为 *x* 的位置。
- y=yval 将列 yval 映射为 *y* 的位置。

在我们赋给 ggplot() 数据框和图形属性映射后，还缺少一个关键的组件：我们需要通知它将哪种几何对象放在那里。此时，ggplot2 尚不知道我们是希望将条形、线条、点还是其他东西绘制到图形上。我们将添加 geom_point() 来绘制点，得到的结果是一幅散点图（见图 A-7）：

```
ggplot(dat, aes(x=xval, y=yval)) + geom_point()
```

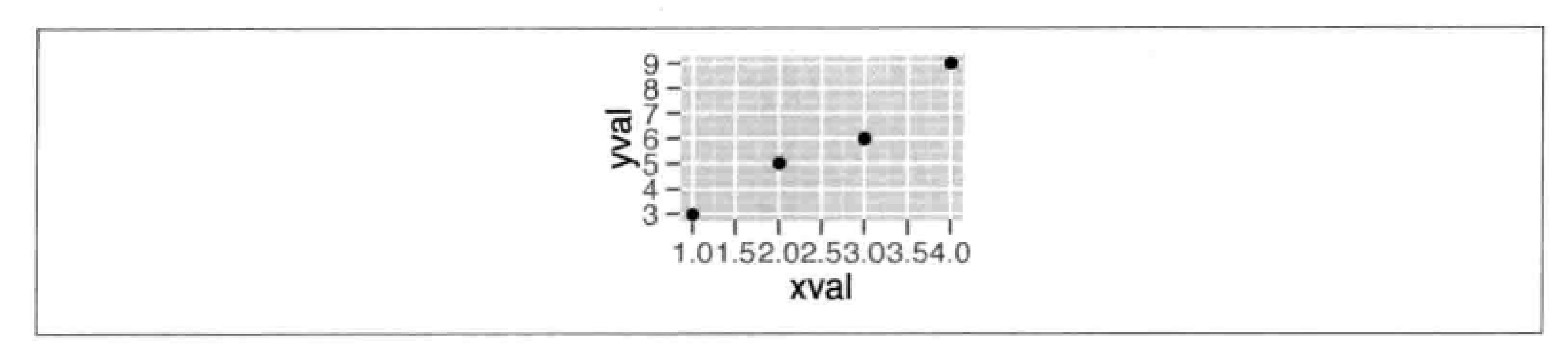


图 A-7 基本的散点图

如果想重用其中的一些组件，可以将其存储到变量中。我们可以把 ggplot 对象保存到 p 中，然后向它添加 geom_point()。这样做的效果与之前的代码相同：

```
p <- ggplot(dat, aes(x=xval, y=yval))
p + geom_point()
```

我们也可以通过将 aes() 放入 geom_point() 的调用中以将变量 group 映射为点的颜色，并指定 colour=group（见图 A-8）：

```
p + geom_point(aes(colour=group))
```

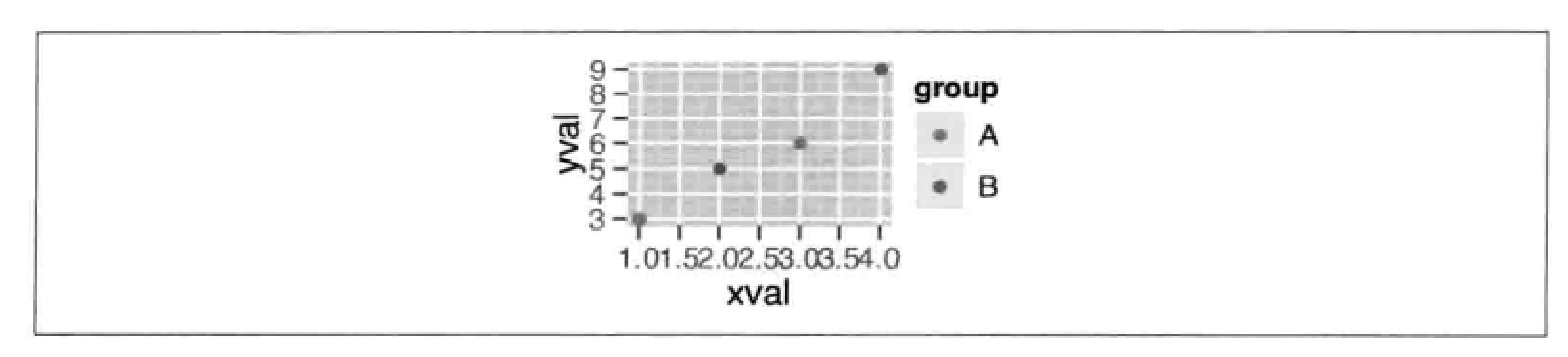


图 A-8 将一个变量映射为颜色的散点图

这样做并不会改变我们之前在 ggplot(...) 内部定义的默认图形属性映射。这样只是为特定的几何对象，即 geom_point() 添加了一个图形属性映射。如果我们再添加其他的几何对象，此映射将不对它们适用。

与这种图形属性映射形成对比的是图形属性的设置。这次，我们不用 aes()；我们将直接设置 colour 的值（见图 A-9）：

```
p + geom_point(colour="blue")
```

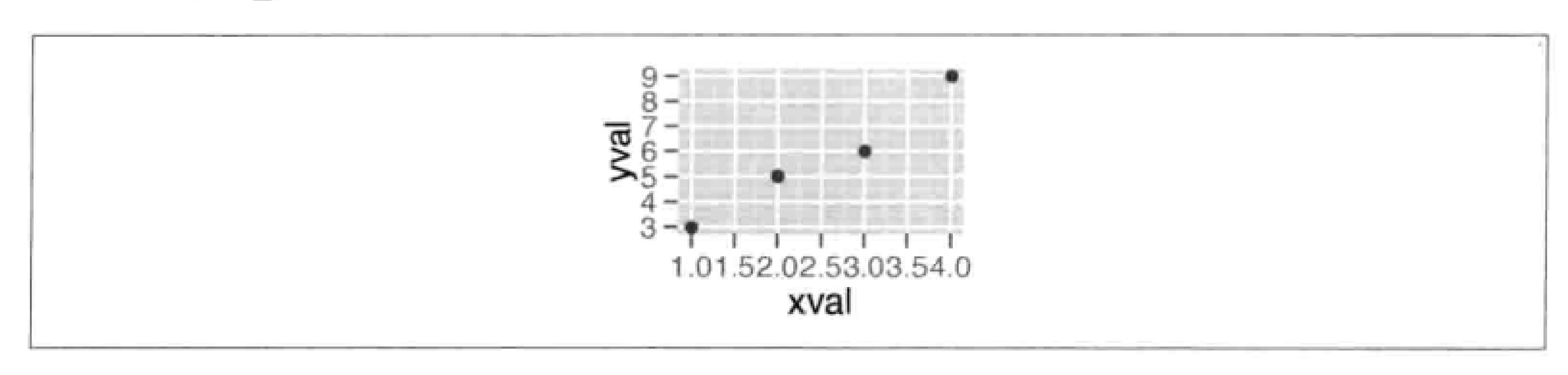


图 A-9　设置了颜色的散点图

我们也可以修改标度，也就是从数据到视觉属性的映射。在此，我们将改变 *x* 标度使其拥有一个更大的范围（见图 A-10）：

```
p + geom_point() + scale_x_continuous(limits=c(0,8))
```

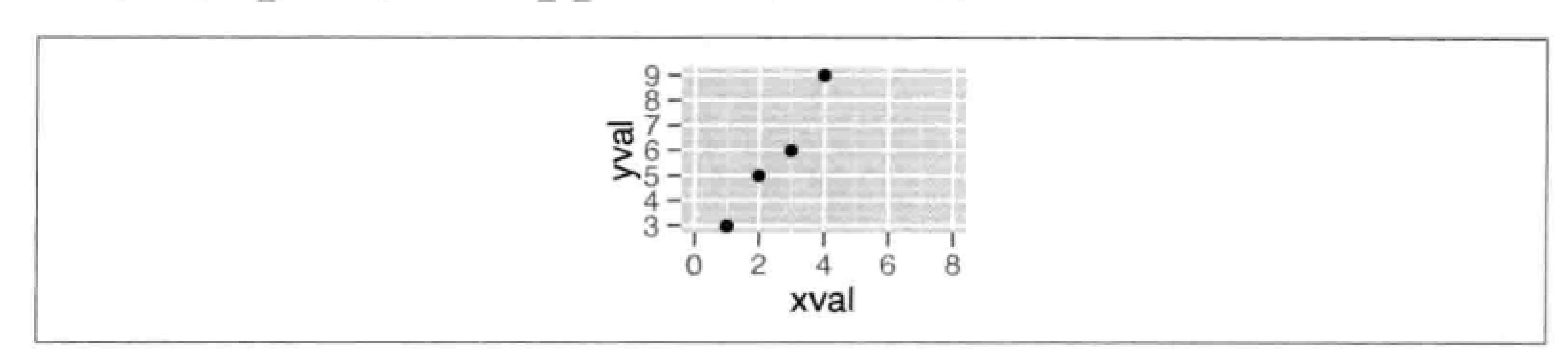


图 A-10　增加了 *x* 范围的散点图

如果回顾一下 colour=group 映射的例子，我们也可以修改颜色标度（见图 A-11）：

```
p + geom_point() +
    scale_colour_manual(values=c("orange","forestgreen"))
```

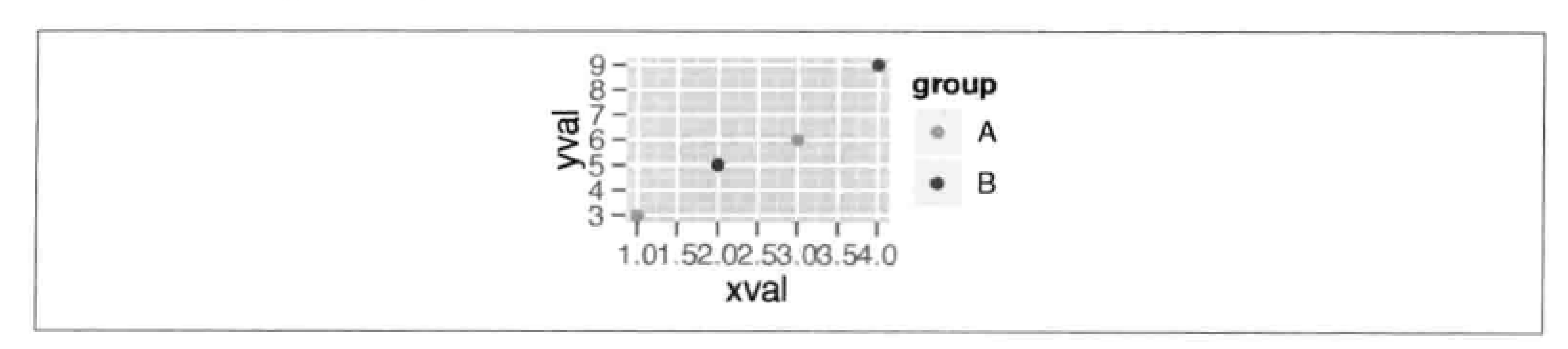


图 A-11　有着修改过的颜色和不同调色板的散点图

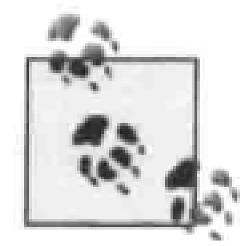
每当我们修改了标度以后，引导元素也发生了变化。对于 *x* 标度，引导元素为 *x* 轴上的刻度线。对于颜色标度，引导元素则为图例。

注意，我们使用了 + 来将每部分连接到一起。在上例中，我们以 + 结束一行，然后在下一行添加更多内容。如果你要写多行，则必须将 + 放在每行的末尾，而不是放在下一行的开始。否则，R 解析器将无法得知会有更多语句出现；它会认为表达式已经结束并对其求值。

A.4 打印输出

在 R 基础图形中，绘图函数会告诉 R 将图形绘制到输出设备（屏幕或者文件）上。ggplot2 略有不同：所用的命令并不直接将图形绘制到输出设备上。相反，函数用于构造图形对象，只有在像 `print(object)` 这样使用 `print()` 函数时图形才会被绘制。你可能会想，“等等，我还没告诉 R 打印任何东西呢，它自己就画出了这些图啊！”然而，这不尽然正确。在 R 中，当你在命令提示符下键入命令以后，实际上发生了两件事情：首先，这条命令得以执行，然后，针对这条命令的返回结果又执行了 `print()`。

交互式 R 提示符下的表现与执行一个脚本或函数时的表现有所不同。在脚本中，命令不会被自动打印出来。函数的情况相同，但是有一个小陷阱：函数中最后一个命令的结果将被返回，所以如果你从 R 提示符中调用函数，最后一个命令的结果将被打印出来，因为它是整个函数的执行结果。



某些 ggplot2 的介绍会使用一个叫做 `qplot()` 的函数，它可以作为一种方便的接口来绘制图形。使用它确实比使用 `ggplot()` 加一个几何对象需要更少的键入，但我发现它用起来有点混乱，因为它指定特定绘图参数的方式有些许不同。我认为还是直接使用 `ggplot()` 更加简单容易。

A.5 统计变换

在将数据映射到图形属性之前，有时必须对其先做变换或汇总。确实如此，例如对于直方图来说，样本点先被分组然后进行计数。每组的计数值用以指定一个条形的高度。有些几何对象，如 `geom_histogram()`，会自动为你做好这些事情，但有些情况下你也会希望使用各种 `stat_xx` 函数来自己做这些事。

A.6 主题

图形外观的某些方面超出了图形语法的范围。其中包括绘图区的背景颜色和网格线、坐标轴标签和图形标题文字使用的字体。这些细节是通过 `theme()` 函数来控制的，已在第 9 章讨论。

A.7 结语

但愿现在的你已经对 ggplot2 幕后的概念有了一定的认识。本书的其余部分将会向你展示如何使用它！

关于作者

Winston Chang 是一名软件工程师，就职于 RStudio 公司，致力于数据可视化及 R 的软件开发工具。他拥有美国西北大学的心理学博士学位。他在读研究生期间，创建了名为“Cookbook for R”的网站，该网站收录了诸多处理 R 软件中常见问题的技巧。在此之前，他曾是哲学研究生并当过程序员。

封面介绍

本书封面上的动物是“驯鹿”（学名角鹿），在北美地区称为北美驯鹿，是栖息于北极和亚北极地区的一个鹿种。驯鹿堪称是为恶劣、寒冷的环境而生，因为它们的皮毛、茸角、嗅觉、蹄子和视力已经全然适应了低温环境。

驯鹿的皮毛由一个长满直立、中空、管状毛的外部皮层和充满类羊毛的底部皮层组成。前者能较好地隔离冷空气并提供水中的浮力。这种皮毛的绝热效果极佳，当驯鹿蜷卧在雪地上时，能不令积雪融化。驯鹿是唯一一个雌雄个体（包括鹿仔）皆长有茸角的鹿种，在现存的所有鹿种中，相对于体型而言，驯鹿拥有的茸角最大。它的茸角每年脱落一次，新茸角会在转年的春天和夏天长出来。

驯鹿的蹄子随季节而变：夏季，苔原柔软湿润，蹄垫会变成类海绵状并提供额外的奔跑动力；冬季，蹄垫会收缩变紧，显露出蹄子的边缘，驯鹿利用这些边缘抓进冰层和积雪层以防止打滑。这也使驯鹿能够从积雪中挖出（常说的挖坑）它们想吃的食物——一种名为石蕊的青苔（即驯鹿苔）。

2012 年，英国伦敦大学学院的研究者发现，驯鹿是唯一一种能够看到紫外线的哺乳动物。人类的视觉只能看到波长在 400 nm 以上的光线，驯鹿的视觉范围则低至 320 nm。虽然这一范围仅仅覆盖了我们基于黑光帮助下所能看见的所有光谱的一部分，但已足以帮助驯鹿在北极的炽白光线中看清东西。如果没有这一点，驯鹿将难以做到。

在圣诞老人的故事里，会飞的驯鹿拉着圣诞老人的雪橇。驯鹿这一意象最早出现在 1823 年一首名为《圣尼古拉斯的来访》的诗中。在这首诗中，驯鹿被称猛冲者（Dasher）、舞者、欢腾、凶婆娘、彗星、丘比特、顿德尔（Dunder）和比利斯克米（Blixem）。

封面图片来自肖的《动物学》。